# UNIVERSIDAD NACIONAL DE LOJA

ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN SISTEMAS

**TÍTULO:**

***"Diseño e implementación de una herramienta visual que permita la creación de nuevos componentes Swing utilizando Java2D y Java3D "***

**Tesis de grado previa a la obtención del Título de Ingenieros en Sistemas.**

**AUTORES:**

**Darwin Augusto Calle Loja**

**Quezada Armijos, José Daniel.**

**DIRECTOR:**

Ing. Wilson Augusto Vélez Ludeña

**Loja – Ecuador**

**2009**

# CERTIFICACIÓN

Sr. Ing.

Wilson Augusto Vélez Ludeña.

**DIRECTOR DE TESIS**

## CERTIFICA:

Que el presente proyecto de tesis elaborado previa la obtención del título en Ingeniería en Sistemas, titulado: “Diseño e Implementación de una Herramienta Visual que permita la creación de nuevos Componentes SWING utilizando Java 2D Y 3D” realizado por los egresados **Darwin Augusto Calle Loja y José Daniel Quezada Armijos**, cumple con los requisitos establecidos por las normas generales para la graduación en la Universidad Nacional de Loja, tanto en aspectos de forma como de contenido; por lo cual me permito autorizar su presentación para los fines pertinentes.

Loja, Junio del 2009

Ing. Wilson Augusto Vélez Ludeña

**DIRECTOR DE TESIS**

# AUTORÍA

Las ideas, conceptos y definiciones expuestas en el presente trabajo de investigación son responsabilidad única de sus autores, ya que están basados en los contenidos recopilados de diversas fuentes bibliográficas, así como de documentos electrónicos de Internet para ponerlos en práctica.

…… …….

**José Daniel Quezada Armijos** **Darwin Augusto Calle Loja**

# DEDICATORIA

Este proyecto lo dedico a mis padres Ramiro Quezada y Fanny Armijos por su paciencia, apoyo y comprensión que me supieron dar en mis días de estudiante y el desarrollo de mi tesis, a mis hermanas, a mis familiares y a todos mis amigos por el apoyo que me brindaron que fue y es un estimulo constante para seguir continuando con mis metas propuestas.

**Daniel**

Con profundo amor y respeto:

A Dios, Ser Supremo creador de todas las cosas.

A mis padres, Cesar Augusto Calle B. y Martha Margarita Loja L por su sacrificio y apoyo constante.

A mis hermanos, soporte fundamental en los duros momentos.

A mis maestros, compañeros y amigos, por brindarme su confianza y amistad durante los años de vida universitaria.

**Darwin**

# AGRADECIMIENTO

Al culminar con éxito esta trabajo, nos es muy grato expresar nuestro más sincero agradecimiento a las autoridades y catedráticos de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, quienes con su significativa labor educativa nos ayudan a descubrir y abrir las puertas del conocimiento, contribuyendo así con la formación profesional en el campo de la informática.

De igual forma, al Ing. Wilson Augusto Vélez Ludeña, Director de Tesis, quien con su acertada dirección y orientación supo guiarnos a la exitosa culminación del presente trabajo investigativo.

Finalmente, a todas y cada una de las personas que con su desinteresado apoyo lograron motivarnos para obtener el resultado final de nuestro esfuerzo, la graduación.

**Los Autores**

INDICE

## ÍNDICE DE FIGURAS

## 1. RESUMEN

El presente trabajo con el tema: ***Diseño e implementación de una herramienta visual que permita la creación de nuevos componentes Swing utilizando Java2D y Java3D*** tiene como su principal objetivo resolver la falta de Herramientas Visuales para mejorar la apariencia gráfica de los controles Swing en las diferentes aplicaciones java realizadas.

El diseño de toda la aplicación está diseñada mediante el paradigma orientado a Objetos, por lo que fué necesario realizar los diagramas de Use Case, Diagrama de Clases, Prototipo de Pantallas, Diagramas de Robustez y Diagramas de Secuencia por cada Use Case, para ello se utilizó la herramienta de modelado UML, Enterprice Architect.

Para la construcción de la aplicación se utilizó como base la máquina virtual de java para la realización de todo el proyecto de tesis, se introdujo al proyecto para la realización de tareas con el S.O Apache-Ant 1.7.0 y para la persistencia de Objetos en java se utilizó XML.

En lo referente al acoplamiento de los componentes generados por la aplicación Swing Evolution se empleó la Tecnología JavaBeans para lograr el reconocimiento dentro de la herramienta NetBeans 5.0 y posteriores.

Como entorno de programación se utilizó NetBeans 6.0, que es un IDE disponible de forma gratuita que ayuda a la facilidad de la programación en el lenguaje Java.

Para la construcción de los nuevos componentes Swing personalizados se utilizó Java2D, el cual contiene un API muy extenso y técnicas de generación de efectos aplicables a los controles Swing. Así mismo con el empleo de esas técnicas de estos APIS se logro implementar los efectos 2D y 3D a estos controles gráficos.

Cabe destacar que esta aplicación está debidamente documentada con los manuales de programador y de usuario para una mejor comprensión, tanto en forma como en diseño.

## 2. INTRODUCCIÓN

En la actualidad el desarrollo de software se hace muy difícil y complejo al tratar de mejorar la apariencia visual de los componentes gráficos SWING de Java, debido al gran tiempo que se toma el programador para su personalización, es así que en algunos casos los usuarios finales (Programadores) tienen la necesidad de obtener una herramienta grafica que les ayuda con su labor.

Las librerías gráficas utilizadas por la plataforma Java en la actualidad únicamente nos brindan una representación gráfica no muy atractiva para muchos quienes la utilizan, de esta manera los desarrolladores de IDE (Entorno Gráfico de Desarrollo) se han propuesto proyectos que ayuden de alguna manera a mejorar la Apariencia Visual de las Interfaces Gráficas de Usuario (GUI) pero hoy en día no existen herramientas que puedan cumplir de manera fácil y sencilla con esta labor.

Un problema que también se presentan muy común en el proceso de la personalización de los componentes SWING es la escritura del código fuente, el mismo que en algunos casos no puede ser entendido de manera rápida por los programadores lo cual les ocasiona confusión en el proceso de aprendizaje.

En la actualidad las herramientas que están manejando los programadores de la plataforma Java se han encargado de la personalización de la GUI (Interfaz Gráfica de Usuario) de manera superficial y básica ,es por ello que vemos la necesidad del Desarrollo de una herramienta de Diseño Gráfico para la personalización de estos componentes Swing.

Para la realización de este trabajo se han planteado los siguientes objetivos:

**Objetivo General**

- Crear de una Herramienta Visual para la personalización de Componentes JFC/SWING

**Objetivos específicos**

- Mejorar el Tiempo de Desarrollo en el modelamiento de los componentes de las aplicaciones Swing con tecnologías Java 2D y 3D.

- Crear una Herramienta de fácil uso al programador para la creación de nuevos Componentes Swing basados en los componentes básicos.

- Permitir la creación de componentes visuales Swing personalizados para el programador.

- Generar el Código fuente en Java de los componentes visuales.

- Lograr el acoplamiento de los nuevos componentes Swing generados por nuestra aplicación en el entorno de desarrollo integrado Netbeans 5.0 y versiones posteriores.

- Implementar la herramienta en la carrera de Ingeniería en Sistemas del Área de Energía, las Industrias y los Recursos Naturales no Renovables.

# 3. METODOLOGÍA

## 3.1. MATERIALES

### 3.1.1. Hardware

- ✓ Computador Portátil.
- ✓ Impresora.
- ✓ Escáner.
- ✓ Memoria Flash 2GB.

### 3.1.2. Software

- ✓ Persistencia de Objetos: XML.
- ✓ Lenguaje de Programación: JAVA.
- ✓ Entorno de programación NetBeans 6.0
- ✓ Herramientas para modelado: Enterprise Architect.
- ✓ Parser XML para la generación Tareas ANT : JDOM
- ✓ Apache-Ant 1.7.0 para tareas de compilación de fuentes Java.

### 3.1.3. Materiales de Oficina

- ✓ Papel
- ✓ Esferográficos
- ✓ Copias
- ✓ Uso de Internet
- ✓ Cartuchos de tinta para impresora.

## 3.2. METODOLOGÍA PARA LA INVESTIGACIÓN

Para el diseño y construcción de la aplicación se realizó el planteamiento del problema, la especificación de requerimientos del sistema y se creó un documento de requerimientos funcionales y no funcionales el cual está redactado en un lenguaje comprensible para las partes (analista y usuario).

Luego se diseñó un modelo de casos de uso, el que permitió comprender de mejor manera los requerimientos que debe cumplir la aplicación y las funcionalidades que este debe realizar.

Inmediatamente se inició con el diseño de aplicación, para lo cual se construyó un diagrama de clases estático y casos de uso de bajo nivel, que consta de los actores del caso de uso, el propósito y descripción del proceso, el curso normal de eventos, que detalla la interacción entre el sistema y los actores. También, se redactó el curso alterno de eventos, que especifica las acciones que forman parte del curso normal, como errores, excepciones u otras opciones.

Seguidamente se elaboró los diagramas de robustez que constituyen el inicio del diseño definitivo de la aplicación; estos diagramas sirven para comprobar que los casos de uso estén correctos y completos. Luego se elaboró los diagramas de secuencia que ayudaron a tener una visión dinámica entre los actores identificados al sistema, las operaciones de éste y las respuestas a los eventos.

A continuación se diseñó un diagrama de clases que sirvió para definir el comportamiento del sistema, es decir, cuales son las clases, los atributos y los métodos que se van a implementar en el diseño de bajo nivel; se lo construyó como una ampliación del modelo conceptual, teniendo en cuenta los diagramas de secuencia y los de robustez.

Para realizar la construcción de la aplicación, se llevó a la práctica todo lo diseñado en fases anteriores. En la construcción de la aplicación se codificó el diseño mediante la

plataforma NetBeans. Posteriormente se realizó un plan de prueba para la aplicación, que nos permitió determinar el correcto o incorrecto funcionamiento de la aplicación.

# 4. FUNDAMENTACIÓN TEÓRICA.

## 4.1 Plataforma Java.-

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que si hacemos un programa en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

## 4.2 Historia de Java.-

**¿Por qué se diseñó Java?**

Los lenguajes de programación C y Fortran se han utilizado para diseñar algunos de los sistemas más complejos en lenguajes de programación estructurada, creciendo hasta formar complicados procedimientos. De ahí provienen términos como “código de espagueti” o “canguros” referentes a programas con múltiples saltos y un control de flujo difícilmente trazable.

No sólo se necesitaba un lenguaje de programación para tratar esta complejidad, sino un nuevo estilo de programación. Este cambio de paradigma de la programación estructurada a la programación orientada a objetos, comenzó hace 30 años con un lenguaje llamado Simula67.

El lenguaje C++ fue un intento de tomar estos principios y emplearlos dentro de las restricciones de C. Todos los compiladores de C++ eran capaces de compilar programas de C sin clases, es decir, un lenguaje capaz de interpretar dos estilos diferentes de programación. Esta compatibilidad (*hacia atrás*) que habitualmente se vende como una característica de C++ es precisamente su punto más débil. No es necesario utilizar un diseño orientado a objetos para programar en C++, razón por la que muchas veces las aplicaciones en este lenguaje no son realmente orientadas al objeto, perdiendo así los beneficios que este paradigma aporta.

## 4.3 Comienzos.-



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Para apreciar el significado e importancia de Java, es muy importante conocer su lugar de origen y cuáles fueron sus propósitos:

En Diciembre de 1990, Patrick Naughton, un empleado de la empresa Sun, reclutó a sus colegas James Gosling y Mike Sheridan para trabajar sobre un nuevo tema conocido como "El proyecto verde". Este a su vez estaba auspiciado por la compañía "Sun founder Bill Joy" y tenía como objetivo principal crear un lenguaje de programación accesible, fácil de aprender y de usar, que fuera universal, y que estuviera basado en un ambiente C++ ya que había mucha frustración por la complejidad y las limitaciones de los lenguajes de programación existentes.

En abril de 1991, el equipo decidió introducir sistemas de software con aplicaciones para consumidores smart como plataforma de lanzamiento para su proyecto. James Gosling escribió el compilador original y lo denominó "Oak", y con la ayuda de los otros miembros del equipo desarrollaron un decodificador que más tarde se convertiría en lenguaje Java.

Para 1992, el equipo ya había desarrollado un sistema prototipo conocido como "*7", que era una especie de cruce entre un asistente digital personalizado y un mecanismo inteligente de control remoto.

Por su parte el presidente de la compañía Sun, Scott McNealy, se dio cuenta en forma muy oportuna y estableció el Proyecto Verde como una subsidiaria de Sun. De 1993 a 1994, el equipo de Naughton se lanzó en busca de nuevas oportunidades en el mercado, mismas que se fueron dando mediante el sistema operativo base.
La incipiente subsidiaria fracasó en sus intentos de ganar una oferta con Time-Warner, sin embargo el equipo concluyó que el mercado para consumidores electrónicos smart y las cajas Set-Up en particular, no eran del todo eficaces.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

La subsidiaria Proyecto Verde fue amortizada por la compañía Sun a mediados del 94'.

Afortunadamente, el cese del Proyecto Verde coincidió con el nacimiento del fenómeno mundial Web. Al examinar las dinámicas de Internet, lo realizado por el ex equipo verde se adecuaba a este nuevo ambiente ya que cumplía con los mismos requerimientos de las set-top box OS que estaban diseñadas con un código de plataforma independiente pero sin dejar de ser pequeñas y confiables.

Patrick Naugthon procedió a la construcción del lenguaje de programación Java que se accionaba con un browser prototipo, más tarde se le fueron incorporando algunas mejoras y el browser Hot Java fue dado a conocer al mundo en 1995.

Con el paso del tiempo el Hot Java se convirtió en un concepto práctico dentro del leguaje Java y demostró que podría proporcionar una forma segura multiplataforma para que el código pueda ser bajado y corrido del Host del World Wide Web y que de otra forma no son seguros.

Una de las características más atractivas del Hot Java fue su soporte para los "applets", que son las partes del código Java que pueden ser cargadas mediante una red de trabajo para después ejecutarlo localmente y así lograr o alcanzar soluciones dinámicas en computación acordes al rápido crecimiento del ambiente Web.

Para dedicarse al desarrollo de productos basados en la tecnología Java, Sun formó la empresa Java Soft en enero de 1996, de esta forma de se dio continuidad al fortalecimiento del programa del lenguaje Java y así trabajar con terceras partes para crear aplicaciones, herramientas, sistemas de plataforma y servicios para aumentar las capacidades del lenguaje.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Durante ese mismo mes, Java Soft dio a conocer el Java Developmet Kit (JDK) 1.0, una rudimentaria colección de componentes básicos para ayudar a los usuarios de software a construir aplicaciones de Java. Dicha colección incluía el compilador Java, un visualizador de applets, un debugger prototipo y una máquina virtual Java (JVM), necesaria para correr programas basados en Java, también incluía paquetería básica de gráficos, sonido, animación y trabajo en red.

Así mismo el Netscape Comunications Inc., mostró las ventajas de Java y rápidamente se asoció con Java Soft para explotar su nueva tecnología. No pasó mucho tiempo antes de que Netscape Communications decidiera apoyar a los Java applets en Netscape Navigator 2.0. Este fue el factor clave que lanzó a Java a ser reconocido y famoso, y que a su vez forzó a otros vendedores para apoyar el soporte de applets en Java.

Como parte de su estrategia de crecimiento mundial y para favorecer la promoción de su nueva tecnología, Java Soft otorgó permisos a otras compañías para que pudieran tener acceso al código fuente de Java y al mismo tiempo mejorar sus navegadores, dicha licencia también les permitía crear herramientas de desarrollo para programación Java y los facultaba para acondicionar Máquinas Virtuales Java (JVM), a varios sistemas operativos.

Muy pronto las licencias o permisos contemplaban a prestigiadas firmas como IBM, Microsoft, Symantec, Silicon Graphics, Oracle, Toshiba y por supuesto Novell.

Desde su aparición, Java se ha ganado una impresionante cantidad de apoyo. Virtualmente cada vendedor importante de software ha obtenido autorización de Java y ahora ha sido incorporado en los principales sistemas operativos base de PC's de escritorio hasta estaciones de trabajo UNIX.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Los nuevos proyectos de Java son co-patrocinados por cientos de millones de dólares en capital disponible de recursos tales como la Fundación Java, un fondo común de capital formado el verano pasado por 11 compañías, incluyendo Cisco Systems, IBM, Netscape y Oracle.

En un reciente estudio se encontró que el 60% de los empresarios están usando Java y el 40% expresaron que Java representa la solución estratégica que estaban buscando para sus negocios.

Para darse una idea de la rápida aceptación que tiene Java en el mundo, tenemos el ejemplo de las conferencias "Java Soft Java One" en San Francisco, el primer Java One fue celebrado en abril de 1996 y atrajo a 5000 usuarios, un año después, en la segunda conferencia Java One albergó a 10,000 usuarios, asistentes. Java Soft estima que el número actual de usuarios Java llega a 400 mil y sigue creciendo. Java también está ganando aceptación en el área empresarial, se ha estimado que actualmente las compañías de hoy que cuentan con más de 5000 empleados, una tercera parte están usando Java.

**Tres de las principales razones que llevaron a crear Java son:**

1. Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.

2. Fiabilidad del código y facilidad de desarrollo. Gosling observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran coste de pruebas y depuración. Por ello en los sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

que encontraba en C++.

3. Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos.

Por todo ello, en lugar de tratar únicamente de optimizar las técnicas de desarrollo y dar por sentada la utilización de C o C++, el equipo de Gosling se planteó que tal vez los lenguajes existentes eran demasiado complicados como para conseguir reducir de forma apreciable la complejidad de desarrollo asociada a ese campo. Por este motivo, su primera propuesta fue idear un nuevo lenguaje de programación lo más sencillo posible, con el objeto de que se pudiese adaptar con facilidad a cualquier entorno de ejecución.

Basándose en el conocimiento y estudio de gran cantidad de lenguajes, este grupo decidió recoger las características esenciales que debía tener un lenguaje de programación moderno y potente, pero eliminando todas aquellas funciones que no eran absolutamente imprescindibles.

## 4.4 Algunas Características.-

Entre las características que nombramos nos referimos a la robustez. Justamente por la forma en que está diseñado, Java no permite el manejo directo del hardware ni de la memoria (inclusive no permite modificar valores de punteros, por ejemplo); de modo que se puede decir que es virtualmente imposible colgar un programa Java. El intérprete siempre tiene el control.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Inclusive el compilador es suficientemente inteligente como para no permitir un montón de cosas que podrían traer problemas, como usar variables sin inicializarlas, modificar valores de punteros directamente, acceder a métodos o variables en forma incorrecta, utilizar herencia múltiple, etc.

Además, Java implementa mecanismos de seguridad que limitan el acceso a recursos de las máquinas donde se ejecuta, especialmente en el caso de los Applets (que son aplicaciones que se cargan desde un servidor y se ejecutan en el cliente).

También está diseñado específicamente para trabajar sobre una red, de modo que incorpora objetos que permiten acceder a archivos en forma remota (vía URL por ejemplo). Además, con el JDK (Java Development Kit) vienen incorporadas muchas herramientas, entre ellas un generador automático de documentación que, con un poco de atención al poner los comentarios en las clases, crea inclusive toda la documentación de las mismas en formato HTML.

### 4.4.1. Independiente de la Plataforma.-

Esto es casi del todo cierto. En realidad, Java podría hacerse correr hasta sobre una Commodore 64. La realidad es que para utilizarlo en todo su potencial, requiere un sistema operativo multithreading (como Unix, Windows95, OS/2).

¿Cómo es esto? Porque en realidad Java es un lenguaje interpretado al menos en principio. Al compilar un programa Java, lo que se genera es un pseudocódigo definido por Sun, para una máquina genérica. Luego, al correr sobre una máquina dada, el software de ejecución Java simplemente interpreta las instrucciones,



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

emulando a dicha máquina genérica. Por supuesto esto no es muy eficiente, por lo que tanto Netscape como Hotjava o Explorer, al ejecutar el código por primera vez, lo van compilando (mediante un *JIT*: *J*ust *&* *T*ime compiler), de modo que al crear por ejemplo la segunda instancia de un objeto el código ya esté compilado específicamente para la máquina huésped.

### 4.4.2.- AWT.-

La Abstract Window Toolkit (AWT, en español Kit de Herramientas de Ventana Abstracta) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java. AWT es ahora parte de las Java Foundation Classes (JFC) - la API estándar para suministrar una interfaz gráfica de usuario (GUI) para un programa Java.

Cuando Sun Microsystems liberó Java en 1995, AWT suministró solo un nivel de abstracción muy fino sobre la interfaz de usuario nativa subyacente. Por ejemplo, crear una caja de verificación AWT causaría que AWT directamente llame a la subrutina nativa subyacente que cree una caja de verificación. Sin embargo, una caja de verificación en Microsoft Windows no es exactamente lo mismo que una caja de verificación en Mac OS o en los distintos tipos de UNIX.

Algunos desarrolladores de aplicaciones prefieren este modelo porque suministra un alto grado de fidelidad al kit de herramientas nativo subyacente y mejor integración con las aplicaciones nativas. En otras palabras, un programa GUI escrito usando AWT parece como una aplicación nativa Microsoft Windows cuando se ejecuta en Windows, pero el mismo programa parece una aplicación nativa Apple Macintosh cuando se ejecuta en un Mac, etc. Sin embargo, algunos desarrolladores de aplicaciones desprecian este modelo porque



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

prefieren que sus aplicaciones se vean exactamente igual en todas las plataformas.

AWT continúa suministrando el núcleo del subsistema de eventos GUI y la interfaz entre el sistema de ventanas nativo y la aplicación Java, suministrando la estructura que necesita Swing. También suministra gestores de disposición básicos, un paquete de transferencia de datos para uso con el Block de notas y Arrastrar y Soltar, y la interface para los dispositivos de entrada tales como el ratón y el teclado.

## 4.5 Swing.-

### 4.5.1. Los Componentes Swing y el Árbol de Contenidos.-

Esta sección presenta algunos de los componentes más utilizados de Swing y explica como los componentes de un GUI entran juntos en un contenedor. Para ilustrarlo, usaremos el programa **SwingApplication** presentado en Una Ruta Rápida por el Código de una Aplicación Swing. Y aquí está su aspecto de nuevo:


**Figura 1.- Componente Swing.**

SwingApplication crea cuatro componentes Swing muy utilizados:

- Una Ventana llamada frame en Swing: **JFrame.**



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

- Un panel llamado en Swing: **JPanel.**
- Un Botón llamado en Swing: **JButton. etc.**

El frame es un contenedor de alto nivel. Existe principalmente para proporcionar espacio para que se dibujen otros componentes Swing. Los otros contenedores de alto nivel más utilizados son los diálogos (**JDialog**) y los applets (**JApplet**).

El panel es un **contenedor intermedio**. Su único propósito es simplificar el posicionamiento del botón y la etiqueta. Otros contenedores intermedios, como los paneles desplazables, (**JScrollPane**) y los paneles con pestañas (**JTabbedPane**), típicamente juegan un papel más visible e interactivo en el GUI de un programa.

El botón y la etiqueta son **componentes atómicos** componentes que existen no para contener otros componentes Swing, sino como entidades auto-suficientes que representan bits de información para el usuario. Frecuentemente, los componentes atómicos también obtienen entrada del usuario. El API Swing proporciona muchos componentes atómicos, incluyendo combo boxes (**JComboBox**), campos de texto (**JTextField**), y tablas (**JTable**).

Aquí podemos ver un diagrama con el **árbol de contenidos** de la ventana mostrada por **SwingApplication**. Este diagrama muestra todos los contenedores creados o usados por el programa, junto con los componentes que contienen. Observa que si añadimos una ventana por ejemplo, un diálogo la nueva ventana tendría su propio árbol de contenidos, independiente del mostrado en esta **Figura 2**.

Como muestra la figura, incluso el programa Swing más sencillo tiene múltiples niveles en su árbol de contenidos. La raíz del árbol de contenidos es siempre un



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

contenedor de alto nivel. Este contenedor proporciona espacio para que sus componentes Swing descendentes se dibujen a sí mismo.

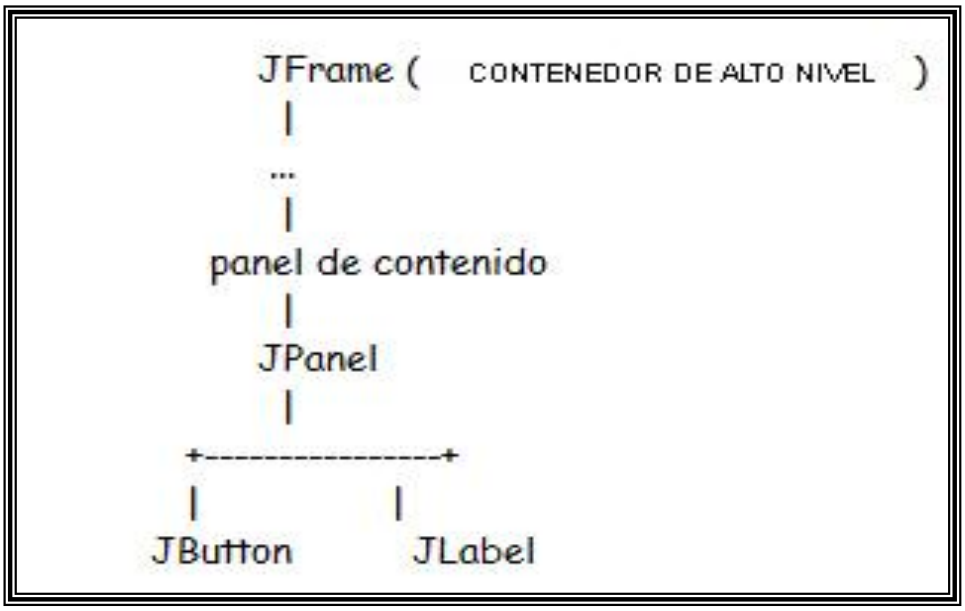



**Figura 2.- Árbol contenidos aplicación Swing.**

Todo contenedor de alto nivel contiene indirectamente un contenedor intermedio conocido como **panel de contenido**. Para la mayoría de los programas no necesitas saber qué pasa entre el contenedor de alto nivel y su panel de contenido.

Cómo regla general, el panel de contenido contiene, directamente o indirectamente, todos los componentes visibles en el GUI de la ventana. La gran excepción a esta regla es que si el contenedor de alto nivel tiene una barra de menú, entonces ésta se sitúa en un lugar especial fuera del panel de contenido.

Para añadir componentes a un contenedor, se usa una de las distintas formas del método **add**. Este método tiene al menos un argumento el componente a añadir. Algunas veces se requiere un argumento adicional para proporcionan información de distribución. Por ejemplo, la última línea del



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

siguiente código de ejemplo especifica que el panel debería estar en el centro de su contenedor (el panel de contenido).

```
frame = new JFrame(...);
button = new JButton(...);
label = new JLabel(...);
pane = new JPanel();
pane.add(button);
pane.add(label);
frame.getContentPane().add(pane, BorderLayout.CENTER);
```

### 4.5.2. Más Características y Conceptos de Swing.-

Esta lección ha explicado algunos de los mejores conceptos que necesitarás conocer para construir GUI Swing el árbol de contenidos, el control de distribución, el manejo de eventos, el dibujado, y los threads. Además, hemos tocado tópicos relacionados, como los bordes. Esta sección explica algunas características Swing que no se han explicado todavía.

### 4.5.3. Características que proporciona JComponent.-

Excepto los contenedores de alto nivel, todos los componentes que empiezan con **J** descienden de la clase **JComponent**. Obtienen muchas características de esta clase, como la posibilidad de tener bordes, tooltips, y Aspecto y Comportamiento configurable. También heredan muchos métodos de conveniencia.

***Iconos.-***



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Muchos componentes Swing principalmente los botones y las etiquetas pueden mostrar imágenes. Estas imágenes se especifican como objetos Icon.

***Actions.-***

Con objetos **Action**, el API Swing proporciona un soporte especial para compartir datos y estados entre dos o más componentes que pueden generar eventos action. Por ejemplo, si tenemos un botón y un ítem de menú que realizan la misma función, podríamos considerar la utilización de un objeto **Action** para coordinar el texto, el icono y el estado de activado de los dos componentes.

### 4.5.4. Soporte para Tecnologías Asistidas.-

Las tecnologías asistidas como los lectores de pantallas pueden usar el API de accesibilidad para obtener información sobre los componentes Swing. Incluso si no hacemos nada, nuestro programa Swing probablemente funcionará correctamente con tecnologías asistidas, ya que el API de accesibilidad está construido internamente en los componentes Swing. Sin embargo, con un pequeño esfuerzo extra, podemos hacer que nuestro programa funcione todavía mejor con tecnologías asistidas, lo que podría expandir el mercado de nuestro programa.

### 4.5.5. Modelos de Datos y Estados Separados.-

La mayoría de los componentes Swing no contenedores tienen modelos. Por ejemplo, un botón (**JButton**) tiene un modelo (**ButtonModel**) que almacena el estado del botón cuál es su menónico de teclado, si está activado, seleccionado o pulsado, etc. Algunos componentes tienen múltiples modelos. Por ejemplo, una lista (**JList**) usa un **ListModel** que almacena los contenidos de la lista y un **ListSelectionModel** que sigue la pista de la selección actual de la lista.

Normalmente no necesitamos conocer los modelos que usa un componente. Por ejemplo, casi todos los programas que usan botones tratan directamente con el objeto **JButton**, y no lo hacen en absoluto con el objeto **ButtonModel**.

Entonces ¿Por qué existen modelos separados? Porque ofrecen la posibilidad de trabajar con componentes más eficientemente y para compartir fácilmente datos y estados entre componentes. Un caso común es cuando un componente, como una lista o una tabla, contiene muchos datos. Puede ser mucho más rápido manejar los datos trabajando directamente con un modelo de datos que tener que esperar a cada petición de datos al modelo. Podemos usar el modelo por defecto del componente o implementar uno propio.

### 4.5.6. El Dibujado en Swing.-

Sin embargo, si tus componentes parece que no se dibujan correctamente, entender los conceptos de esta sección podría ayudarte a ver qué hay erróneo. De igual modo, necesitarás entender esta sección si creas código de dibujo personalizado para un componente.

***Cómo funciona el dibujado.-***

Cuando un GUI Swing necesita dibujarse a sí mismo la primera vez, o en respuesta a la vuelta de un ocultamiento, o porque necesita reflejar un cambio en el estado del programa empieza con el componente más alto que necesita ser redibujado y va bajando por el árbol de contenidos. Esto está orquestado por el sistema de dibujo del AWT, y se ha hecho más eficiente mediante el manejador de dibujo de Swing y el código de doble buffer.

Los componentes Swing generalmente se redibujan a sí mismos siempre que es necesario. Por ejemplo, cuando llamamos al método **setText** de un componente, el componente debería redibujarse automáticamente a sí mismo, y si es necesario, redimensionarse. Si no lo hace así es un bug. El atajo es



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

llamar al método **repaint** sobre el componente para pedir que el componente se ponga en la cola para redibujado. Si se necesita cambiar el tamaño o la posición del componente pero no automáticamente, deberíamos llamar al método **revalidate** sobre el componente antes de llamar a **repaint**.

Al igual que el código de manejo de eventos, el código de dibujo se ejecuta en el thread del despacho de eventos. Mientras se esté manejando un evento no ocurrirá ningún dibujo. De forma similar, si la operación de dibujado tarda mucho tiempo, no se manejará ningún evento durante ese tiempo.

Los programas sólo deberían dibujarse cuando el sistema de dibujo se lo diga. La razón es que cada ocurrencia de dibujo de un propio componente debe ser ejecutado sin interrupción. De otro modo, podrían ocurrir resultados impredecibles: como que un botón fuera dibujado medio pulsado o medio liberado.

Para acelerar, el dibujo Swing usa **doble-buffer** por defecto realizado en un buffer fuera de pantalla y luego lanzado a la pantalla una vez finalizado. Podría ayudar al rendimiento si hacemos un componente Swing opaco, para que el sistema de dibujo de Swing pueda conocer lo que no tiene que pintar detrás del componente. Para hacer opaco un componente Swing, se llama al método **setOpaque (true)** sobre el componente.

Los componentes no-opacos de Swing puede parecer que tienen cualquier forma, aunque su área de dibujo disponible es siempre rectangular. Por ejemplo, un botón podría dibujarse a sí mismo dibujando un octógono relleno. El componente detrás del botón, (su contenedor, comúnmente) sería visible, a través de las esquinas de los lados del botón. El botón podría necesitar incluir código especial de detección para evitar que un evento action cuando el usuario pulsa en las esquinas del botón.

***Un Ejemplo de Dibujo***



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

Para ilustrar el dibujado, usaremos el programa **SwingApplication**, que se explicó. Aquí podemos ver el GUI de **SwingApplication**:


**Figura 3.- Ejemplo de Dibujo**

Y aquí su árbol de contenidos:

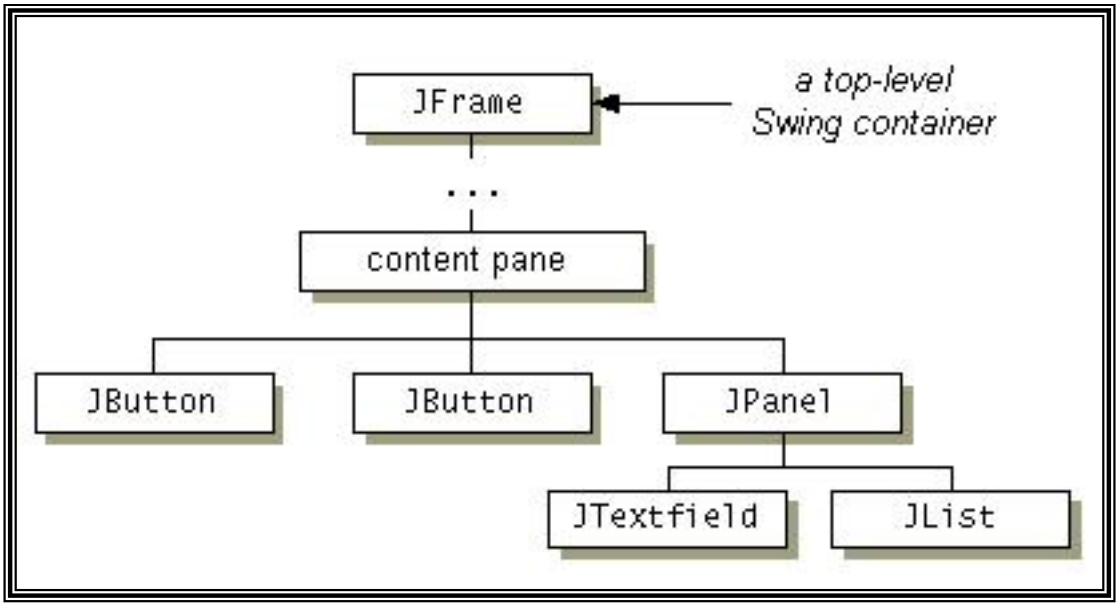



**Figura 4.- Árbol de Contenidos.**

Aquí está lo que sucede cuando se dibuja el GUI de **SwingApplication**:

1. El contenedor de alto nivel, **JFrame**, se dibuja as sí mismo.



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

2. El panel de contenido primero dibuja su fondo, que es un rectángulo sólido de color gris. Luego le dice al **JPanel** que se dibuje el mismo. El rectángulo del panel de contenido realmente no aparece en el GUI finalizado porque está oscurecido por el **JPanel**.

   **Nota:** Es importante que el panel de contenido sea opaco. De otro modo, resultará en dibujados confusos. Cómo el **JPanel** es opaco, podemos hacer que sea el panel de contenido (sustituyendo **setContentPane** por el código existente **getContentPane ().add**). Esto simplifica considerablemente el árbol de contenidos y el dibujado, eliminado un contenedor innecesario.

3. El **JPanel** primero dibuja su fondo, un rectángulo sólido de color gris. Luego dibuja su borde. El borde es un **EmptyBorder**, que no tendrá efecto excepto para incrementar el tamaño del **JPanel** reservando algún espacio extra en los laterales del panel. Finalmente, el panel le pide a sus hijos que se dibujen a sí mismos.
4. Para dibujarse a sí mismo, el **JButton** dibuja su rectángulo de fondo si es necesario y luego dibuja el texto que contiene. Si el botón tiene el foco del teclado, significa que cualqu**ier co**sa que se teclee va directamente al botón para su procesamiento, luego el botón realiza algún dibujado específico del Aspecto y Comportamiento para aclarar que tiene el foco.
5. Para dibujarse a sí misma, la **JLabel** dibuja su texto.

De este modo, cada componente se dibuja a sí mismo antes de que lo haga cualquier componente que contenga, Esto asegura que el fondo de un **JPanel**, por ejemplo, sólo se dibuja cuando no está cubierto por uno de los componentes que contiene. La siguiente figura ilustra el orden en que cada componente que desciende de **JComponent** se dibuja a sí mismo:



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

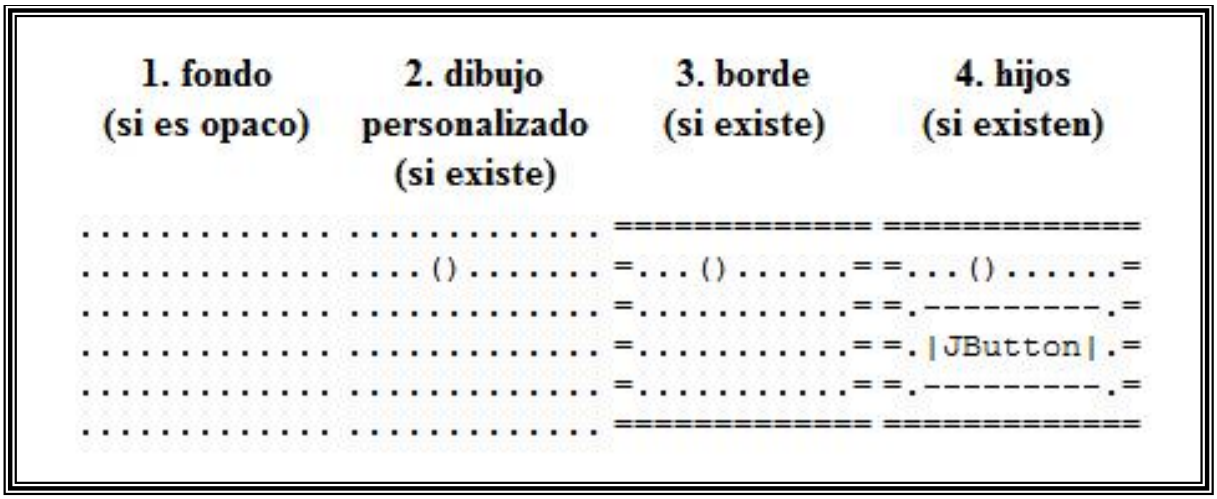



**Figura 5.- Proceso de Redibujado**



Texto Tomado de: http://www.sun.com

Traducción: http://java.sun.com/books/Series/Tutorial/index.html

Año Publicación: 22-10-2006

## 4.6 JAVA 2D.-

### 4.6.1. Sistema de Coordenadas.-

El sistema 2D de Java mantiene dos espacios de coordenadas:

El espacio de usuario es el espacio en que se especifican los gráficos primitivos. El espacio de dispositivo es el sistema de coordenadas para un dispositivo de salida, como una pantalla, una ventana o una impresora.

El espacio de usuario es un sistema de coordenadas lógicas independiente del dispositivo: el espacio de coordenadas que usan nuestros programas. Todos los geométricos pasados a las rutinas Java 2D de renderizado se especifican en coordenadas de espacio de usuario.

Cuando se utiliza la transformación por defecto desde el espacio de usuario al espacio de dispositivo, el origen del espacio de usuario es la esquina superior izquierda del área de dibujo del componente.

La coordenada X se incrementa hacia la derecha, y la coordenada Y hacia abajo.

El espacio de dispositivo es un sistema de coordenadas dependiente del dispositivo que varía de acuerdo a la fuente del dispositivo. Aunque el sistema de coordenadas para una ventana o una pantalla podría ser muy distinto que para una impresora, estas diferencias son invisibles para los programas Java.

Las conversiones necesarias entre el espacio de usuario y el espacio de dispositivo se realizan automáticamente durante el dibujado.



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

### 4.6.2. Construir Formas Complejas desde Gráficos Primitivos.-

Construir un área geométrica (CAG) es el proceso de crear una nueva forma geométrica realizando operaciones con las ya existentes. En el API Java 2D un tipo especial de Shape llamado Área soporta operaciones booleanas. Podemos construir un Área desde cualquier Shape.

Área soporta las siguientes operaciones booleanas:

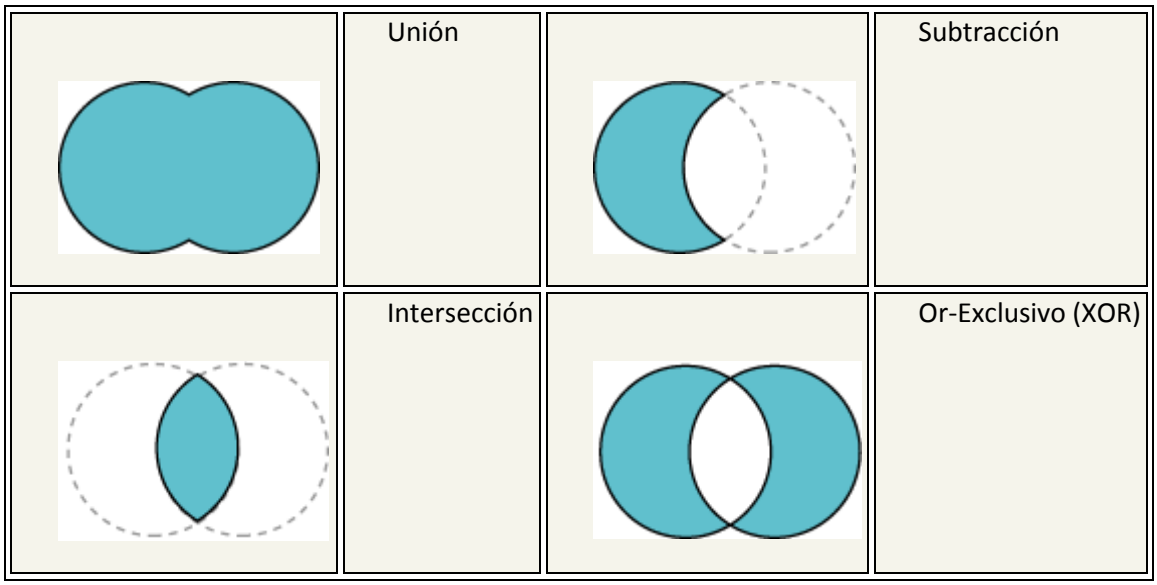



**Figura 6.- Operaciones Lógicas.**

- **Ejemplo: Áreas**



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

En este ejemplo, los objetos Área construyen una forma de pera partiendo de varias elipses.

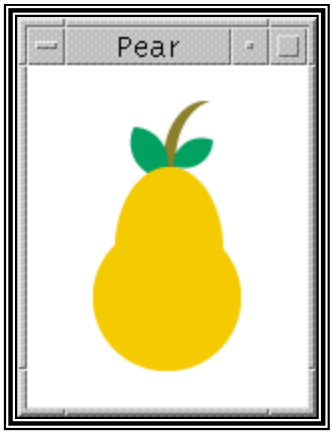



**Figura 7.- Ejemplo de Operaciones Lógicas entre Shapes.**

### 4.6.3. Gráficos en 2 Dimensiones.-

**Gráficos en 2 Dimensiones:**

El API 2D de Java permite fácilmente:

- Dibujar líneas de cualquier anchura
- Rellenar formas con gradientes y texturas
- Mover, rotar, escalar y recortar texto y gráficos.
- Componer texto y gráficos solapados.

Por ejemplo, se podría mostrar gráficos y charts complejos que usan varios estilos de línea y de relleno para distinguir conjuntos de datos, como se muestra en la siguiente figura:

**Figura 8.- Dibujos Complejos.**

El API 2D de Java también permite almacenar datos de imágenes por ejemplo, se puede realizar fácilmente filtros de imágenes, como blur o recortado, como se muestra en la siguiente figura:


**Figura 9.- Proceso de Tratamiento de Imágenes.**

### 4.6.4. Formas 2D.-

Las clases del paquete java.awt.geom definen gráficos primitivos comunes, como puntos, líneas, curvas, arcos, rectángulos y elipses.

Clases en el paquete java.awt.geom.

| | | |
|---|---|---|
| Arc2D | Ellipse2D | QuadCurve2D |



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

| | | |
|---|---|---|
| Area | GeneralPath | Rectangle2D |
| CubicCurve2D | Line2D | RectangularShape |
| Dimension2D | Point2D | RoundRectangle2D |

**Figura 10.- Clases Importantes de paquete java.awt.geom.**

Excepto para Point2D y Dimension2D, cada una de las otras clases geométricas implementa el interface Shape, que proporciona un conjunto de métodos comunes para describir e inspeccionar objetos geométricos bi-dimensionales.

Con estas clases se puede crear de forma virtual cualquier forma geométrica y dibujarla a través de Graphics2D llamando al método draw o al método fill. Por ejemplo, las formas geométricas en la siguiente figura 11 están definidas usando los geométricos básicos de Java 2D.

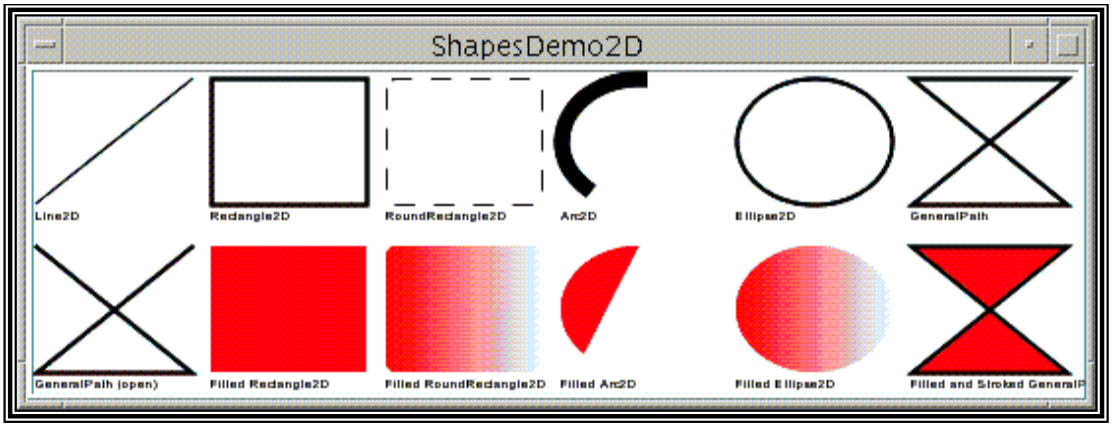



**Figura 11.- Figuras Geométricas.**

- **Formas Rectangulares**



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

Los primitivos Rectangle2D, RoundRectangle2D, Arc2D, y Ellipse2D descienden del RectangularShape, que define métodos para objetos Shape que pueden ser descritos por una caja rectangular. La geometría de un RectangularShape puede ser extrapolada desde un rectángulo que encierra completamente el exterior de la Shape.

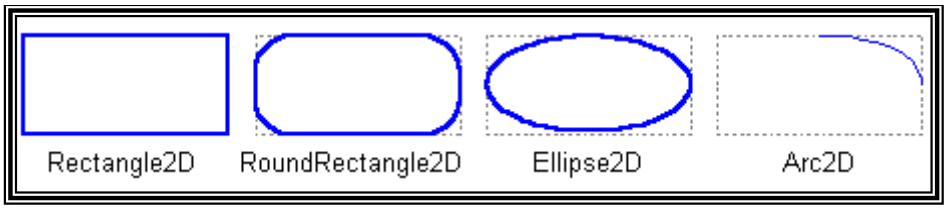



**Figura 12.- Figuras Rectangulares.**

- **GeneralPath**

La clase GeneralPath permite crear una curva arbitraria especificando una serie de posiciones a lo largo de los límites de la forma. Estas posiciones pueden ser conectadas por segmentos de línea, curvas cuadráticas o curvas cúbicas. La siguiente figura puede ser creada con 3 segmentos de línea y una curva cúbica.

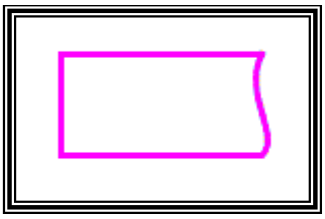


**Figura 13.- Figuras con GeneralPath.**

- **Áreas**

Con la clase Área se realiza operaciones booleanas, como uniones, intersecciones y substracciones, sobre dos objetos Shape cualesquiera. Esta técnica, permite crear rápidamente objetos Shape complejos sin tener que describir cada línea de segmento o cada curva.

### 4.6.5. Controlar la Calidad del Dibujo.-

Se puede usar el atributo 'rendering hint' de Graphics2D para especificar que los objetos sean dibujados tan rápido como sea posible o que se dibujen con la mayor calidad posible.

Para seleccionar o configurar el atributo 'rendering hint' en el contexto, Graphics2D se puede construir un objeto RenderingHints y pasarlo dentro de Graphics2D.setRenderingHints. Si sólo se quiere seleccionar un hint, se llama a Graphics2D. setRenderingHint y especificar la pareja clave-valor para el hint que se desea seleccionar.

RenderingHints soporta los siguientes tipos de hints:

- Alpha interpolation.-
- Antialiasing.-
- Color rendering.-
- Dithering.-
- Fractional Metrics.-
- Interpolation.-
- Rendering.-
- Text Antialiasing.-

### 4.6.6. Renderizado en Java2D.-

El mecanismo de renderizado básico es el mismo que en las versiones anteriores del JDK, el sistema de dibujo controla cuándo y cómo dibuja un programa. Cuando un componente necesita ser mostrado, se llama automáticamente a su método **paint** o **update** dentro del contexto **Graphics** apropiado.

El API 2D de Java presenta java.awt.Graphics2D, un nuevo tipo de objeto **Graphics**. **Graphics2D** desciende de la clase Graphics para proporcionar acceso a las características avanzadas de renderizado del API 2D de Java.

Para usar las características del API 2D de Java, se forza el objeto **Graphics** pasado al método de dibujo de un componente a un objeto **Graphics2D**.

```
public void Paint (Graphics g) {
 Graphics2D g2 = (Graphics2D) g;
}
```

### 4.6.6.1. Contexto de Renderizado de Graphics2D.-

Al conjunto de atributos de estado asociados con un objeto **Graphics2D** se le conoce como **Contexto de Renderizado de Graphics2D**. Para mostrar texto, formas o imágenes, se configura este contexto y luego se llama a uno de los métodos de renderizado de la clase **Graphics2D**, como **draw** o **fill**. Cómo muestra la siguiente figura, el contexto de renderizado de **Graphics2D** contiene varios atributos.



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

| | |
|---|---|
|  | El estilo de lápiz que se aplica al exterior de una forma. Este atributo **stroke** nos permite dibujar líneas con cualquier tamaño de punto y patrón de sombreado y aplicar finalizadores y decoraciones a la línea. |
|  | El estilo de relleno que se aplica al interior de la forma. Este atributo **paint** nos permite rellenar formas con cólores sólidos, gradientes o patrones. |
|  | El estilo de composición se utiliza cuando los objetos dibujados se solapan con objetos existentes. |
|  | La transformación que se aplica durante el dibujado para convertir el objeto dibujado desde el espacio de usuario a las coordenadas de espacio del dispositivo. También se pueden aplicar otras transformaciones opcionales como la traducción, rotación escalado, recortado, a través de este atributo. |
|  | El Clip que restringe el dibujado al área dentro de los bordes de la **Shape** se utiliza para definir el ára de recorte. Se puede usar cualquier **Shape** para definir un clip. |
| | La **fuente** se usa para convertir cadenas de texto. |



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

| | |
|---|---|
| Aa Bb | |
| a Antialiasing | Punto de Renderizado que especifican las preferencias en cuanto a velocidad y calidad. Por ejemplo, podemos especificar si se debería usar antialiasing, si está disponible. |

**Figura 14.- Atributos de Graphics2D.**

Para configurar un atributo en el contexto de renderizado de **Graphics2D**, se usan los métodos **set Attribute**.

- **setStroke.-** Estilo de línea para dibujar.
- **setPaint.-** Contorno de relleno de dibujo, utilizado en áreas.
- **setComposite.-** Permite la realización de composiciones entre figuras.
- **setTransform.-** Permite trasladar, girar, convertir puntos de posición en los objetos a renderizar.
- **setClip.-** Permite obtener una parte de un área de un objeto Shape.
- **setFont.-** Permite fijar un estilo de fuente para la creación de textos.
- **setRenderingHints.-** Permite aplicar algoritmos de renderización a los objetos Graphics2D para una mejor apariencia.

Cuando se configura un atributo, se le pasa al objeto el atributo apropiado. Por ejemplo, para cambiar el atributo paint a un relleno de gradiente azul-gris, se debería construir el objeto **GradientPaint** y luego llamar a **setPaint**.

```
gp = new GradientPaint(0f,0f,blue,0f,30f,green);
```

```
g2.setPaint (gp);
```

**Graphics2D** contiene referencias a sus objetos atributos no son clonados. Si modificamos un objeto atributo que forma parte del contexto **Graphics2D**, necesitamos llamar al método **set** para notificarlo al contexto. La modificación de un atributo de un objeto durante el renderizado puede causar comportamientos impredecibles.

### 4.6.6.2. Contexto de Renderizado draw y fill.-

**Graphics2D** proporciona los siguientes métodos generales de dibujado que pueden usarse para dibujar cualquier primitivo geométrico, texto o imagen.

- **draw**.- Dibuja el exterior de una forma geométrica primitiva usando los atributos stroke y paint.

- **fill**.- Dibuja cualquier forma geométrica primitiva rellenado su interior con el color o patrón especificado por el atributo paint.

- **drawString**.- Dibuja cualquier cadena de texto. El atributo font se usa para convertir la fuente a glyphs que luego se rellenan con el color o patrón especificados por el atributo paint.

- **drawImage**.- Dibuja la imagen especificada.

Además, **Graphics2D** soporta los métodos de renderizado de **Graphics** para formas particulares, como **drawOval** y **fillRect**.

### 4.6.7. Componer Gráficos.-

La clase **AlphaComposite** encapsula varios estilos de composición, que determinan cómo se dibujan los objetos solapados. Un **AlphaComposite** también puede tener un valor alpha que especifica el grado de transparencia: alpha = 1.0 es totalmente opaco, alpha = 0.0 es totalmente transparente. **AlphaComposite** soporta la mayoría de los estándares de composición como se muestra en la siguiente tabla.

| | |
|---|---|
| Source-over (**SRC_OVER**)   | Si los pixels del objeto que está siendo renderizado (la fuente) tienen la misma posición que los pixels renderizados préviamente (el destino), los pixels de la fuente se renderizan sobre los pixels del destino. |
| Source-in (**SRC_IN**)   | Si los pixels de la fuente y el destino se solapan, sólo se renderizarán los pixels que haya en el área solapada. |
| Source-out (**SRC_OUT**)   | Si los pixels de la fuente y el destino se solapan, sólo se renderizarán los pixels que haya fuera del área solapada. Los pixels que haya en el área solapada se borrarán. |
| Destination-over | Si los pixels de la fuente y del destino se solapan, |



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

| | |
|---|---|
| (**DST_OVER**)<br>  <br> | sólo se renderizarán los pixels de la fuente que haya fuera del área solapada. Los pixels que haya en el área solapada no se cambian. |
| Destination-in (**DST_IN**)<br>  <br> | Si los pixels de la fuente y del destino se solapan, el alpha de la fuente se aplica a los pixels del área solapada del destino. Si el alpha = 1.0, los pixels del área solapada no cambian; si alpha es 0.0 los pixels del área solapada se borrarán. |
| Destination-out<br>(**DST_OUT**)<br>  <br> | Si los pixels de la fuente y del destino se solapan, se aplica el alpha de la fuente a los pixels del área solapada del destino. Si el alpha = 1.0, los pixels del área solapada no cambian; si alpha es 0.0 los pixels del área solapada se borrarán. |
| Clear (**CLEAR**)<br>  <br> | Si los pixels de la fuente y del destino se solapan, los pixels del área solapada se borrarán. |

**Figura 15.- Composición de Gráficos.**

Para cambiar el estilo de composición usado por **Graphics2D**, se crea un objeto **AlphaComposite** y se pasa al método **setComposite**.

- **Ejemplo: Composite**

Esta figura ilustra los efectos de varias combinaciones de estilos de composición y valores de alpha.

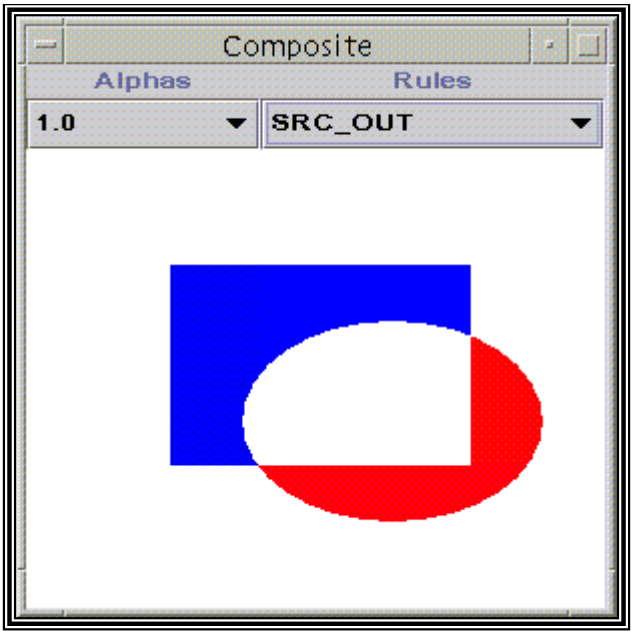



**Figura 16.- Ejemplo de Composite.**

Se ha construido un nuevo objeto **AlphaComposite ac** llamando a **AlphaComposite**. **getInstance** y especificando las reglas de composición deseadas.

```
AlphaComposite ac =  AlphaComposite.getInstance (
                        AlphaComposite.SRC);
```

Cuando se selecciona una regla de composición o un valor alpha, se llama de nuevo a **AlphaComposite.getInstance**, y el nuevo **AlphaComposite** se asigna a **ac**. El alpha seleccionado se aplica al valor alpha de cada pixel y se le pasa un segundo parámetro a **AlphaComposite**.**getInstance**.

```
ac = AlphaComposite.getInstance(getRule(rule), alpha);
```

El atributo composite se modifica pasando el objeto **AlphaComposite** a **Graphics 2D setComposite**. Los objetos son renderizados dentro de un **BufferedImage** y más tarde se copian en la pantalla, por eso el atributo composite se configura con el contexto **Graphics2D** para el **BufferedImage**.

```
BufferedImage buffImg = new BufferedImage (w, h,
                                BufferedImage.TYPE_INT_ARGB)
                                ;

Graphics2D gbi = buffImg.createGraphics ();
gbi.SetComposite(ac);
```

### 4.6.8. Punteado y Relleno de Gráficos Primitivos.-

Cambiando el punteado y los atributos de dibujo en el contexto de Graphics2D, antes del dibujo, podemos fácilmente aplicar estilos divertidos de líneas y patrones de relleno para gráficos primitivos. Por ejemplo, podemos dibujar una línea punteada creando el objeto Stroke apropiado y llamando a setStroke para añadirlo al contexto Graphics2D antes de dibujar la línea. De forma similar, podemos aplicar un relleno de gradiente a un Shape creando un objeto GradientPaint y añadiendo al contexto Graphics2D llamando a setPaint antes de dibujar la Shape.

La siguiente imagen demuestra cómo podemos dibujar formas geométricas usando los métodos Graphics2D draw y fill.

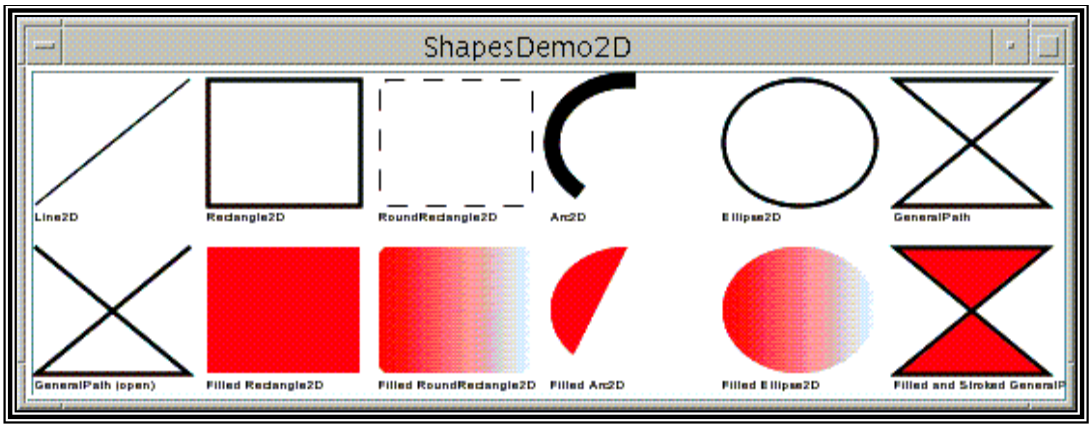
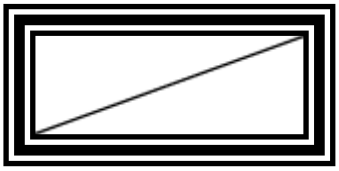



**Figura 17.- Figuras Geométricas, dibujadas con draw y fill.**

Cada una de las formas de la imagen está construida de geometrías y está dibujada a través de Graphics2D. Las variables rectHeight y rectWidth de este ejemplo definen las dimensiones del espacio en que se dibuja cada forma, en pixels. La variables x e y cambian para cada forma para que sean dibujadas en formación de parrilla.



```
// draw Line2D.Double

g2.draw (new Line2D.Double(x, y+rectHeight-1,

                              x +rectWidth,
```

```
// draw Rectangle2D.Double

g2.setStroke (stroke);

g2.draw (new Rectangle2D.Double(x, y,rectWidth
```

```
// draw RoundRectangle2D.Double

g2.setStroke (dashed);

g2.draw (new RoundRectangle2D.Double(x,y,rectWidth,
```

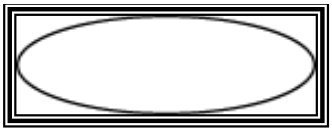


Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

```
// draw Arc2D.Double

g2.setStroke (wideStroke);

g2.draw (new Arc2D.Double(x,y,rectWidth,rectHeight,
```



```
// draw Ellipse2D.Double

g2.setStroke(stroke);

g2.draw (new Ellipse2D.Double(x, y,
```

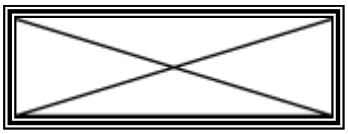
```
// draw GeneralPath (polygon)

int x1Points[] = {x, x+rectWidth,x, x+rectWidth};

int y1Points[] = {y, y+rectHeight,y+rectHeight, y};

GeneralPath polygon = new GeneralPath(

                      GeneralPath.WIND_EVEN_ODD,

                      x1Points.length);

polygon.moveTo (x1Points[0], y1Points[0]);

for (int index = 1;index < x1Points.length;index++)
```



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

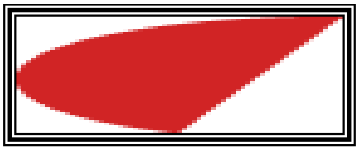
Año Publicación: 05-01-2008



```
// draw GeneralPath (polyline)

int x2Points[] = {x, x+rectWidth, x,x+rectWidth};

int y2Points[] = {y, y+rectHeight,y+rectHeight, y};

GeneralPath polyline = new GeneralPath(

                        GeneralPath.WIND_EVEN_ODD,

                        x2Points.length);

polyline.moveTo (x2Points[0], y2Points[0]);
```

```
// fill Rectangle2D.Double (red)

g2.setPaint (red);

g2.fill (new Rectangle2D.Double(x, y,
```

```
// fill RoundRectangle2D.Double

g2.setPaint(redtowhite);

g2.fill(new RoundRectangle2D.Double(x, y,

                                    rectWidth,
```



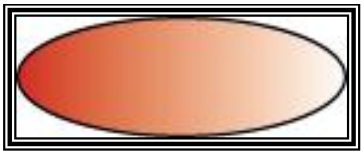
```
// fill Arc2D

g2.setPaint (red);

g2.fill (new Arc2D.Double(x, y, rectWidth,
```



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008



```
// fill Ellipse2D.Double

g2.setPaint (redtowhite);

g2.fill (new Ellipse2D.Double(x, y,
```

```
// fill and stroke GeneralPath

int x3Points[] = {x, x+rectWidth, x,x+rectWidth};

int y3Points[] = {y, y+rectHeight,y+rectHeight, y};



GeneralPath filledPolygon = new GeneralPath(

                                GeneralPath.WIND_EV
                                EN_ODD,

                                x3Points.length);

filledPolygon.moveTo(x3Points[0],y3Points[0]);

for (int index = 1;index < x3Points.length;index++)
{

        filledPolygon.lineTo(x3Points[index],

                                y3Points[index]);
```

**Figura 18.- Ejemplos de draw y fill.**

Observa que este ejemplo usa implementaciones de doble precisión de las clases geométricas. Donde sea posible, las implementaciones de los float y doble precisión de cada geométrico están proporcionadas por clases internas.

### 4.6.9. Definir Estilos de Línea y Patrones de Relleno.-

Probablemente habrás observado en el ejemplo anterior algunas de las formas tienen líneas punteadas o están rellenas con gradientes de dos colores. Usando las clases Stroke y Paint de Java 2D, podemos fácilmente definir estilos de línea divertidos y patrones de relleno.

- **Estilos de Línea**

Los estilos de línea están definidos por el atributo stroke en el contexto Graphics2D. Para seleccionar el atributo stroke podemos crear un objeto BasicStroke y pasarlo dentro del método Graphics2D setStroke.

Un objeto BasicStroke contiene información sobre la anchura de la línea, estilo de uniones, estilos finales, y estilo de punteado. Esta información se usa cuando se dibuja un Shape con el método draw.

La anchura de línea es la longitud de la línea medida perpendicularmente a su trayectoria. La anchura de la línea se especifica como un valor float en las unidades de coordenadas de usuario, que es equivalente a 1/72 pulgadas cuando se utiliza la transformación por defecto.

El estilo de unión es la decoración que se aplica cuando se encuentran dos segmentos de línea. BasicStroke soporta tres estilos de unión:

JOIN_BEVEL

JOIN_MITER

JOIN_ROUND

El estilo de finales es la decoración que se aplica cuando un segmento de línea termina. BasicStroke soporta tres estilos de finalización:

CAP_BUTT

CAP_ROUND

CAP_SQUARE

El estilo de punteado define el patrón de las secciones opacas y transparentes aplicadas a lo largo de la longitud de la línea. Este estilo está definido por un array de punteado y una fase de punteado. El array de punteado define el patrón de punteado. Los elementos alternativos en el array representan la longitud del punteado y el espacio entre punteados en unidades de coordenadas de usuario. El elemento 0 representa el primer punteado, el elemento 1 el primer espacio, etc. La fase de punteado es un desplazamiento en el patrón de punteado, también especificado en unidades de coordenadas de usuario. La fase de punteado indica que parte del patrón de punteado se aplica al principio de la línea.

**Patrón de Relleno**

Los patrones de rellenos están definidos por el atributo paint en el contexto Graphics2D. Para seleccionar el atributo paint, se crea un ejemplar de un objeto que implemente el interface Paint y se pasa dentro del método Graphics2D setPaint.

Tres clases implementan el interface Paint: Color, GradientPaint, y TexturePaint.

Para crear un GradientPaint, se especifica una posición inicial y un color y una posición final y otro color. El gradiente cambia proporcionalmente desde un color al otro a lo largo de la línea que conecta las dos posiciones.

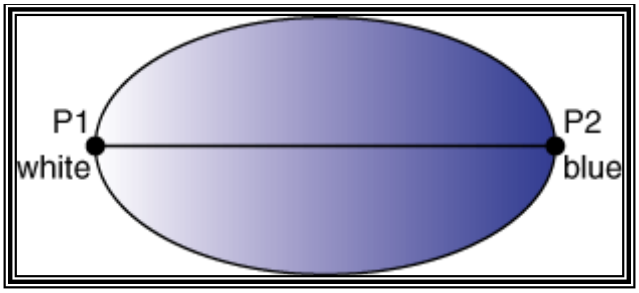



**Figura 19.- Creación de una GradientePaint.**

El patrón para una TexturePaint está definido por un BufferedImage. Para crear un TexturePaint, se especifica una imagen que contiene el patrón y un rectángulo que se usa para replicar y anclar el patrón.

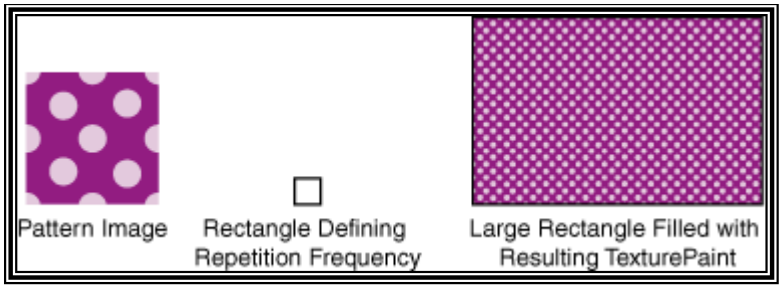



**Figura 20.- Creación de una TexturePaint.**

- Ejemplo: **StrokeAndFill**

En la siguiente figura se observa el estilo de línea, estilo de dibujo y el punteado exterior del objeto.

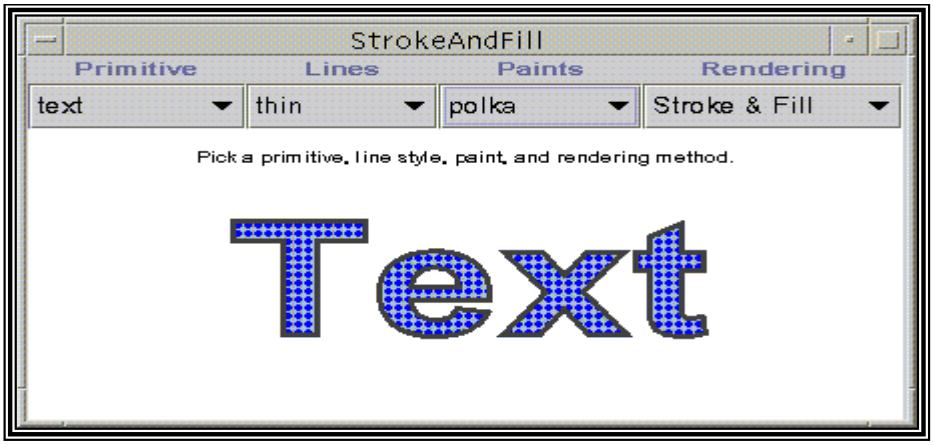



**Figura 21.- Utilización de Stroke & Fill.**

Los primitivos son inicializados e introducidos en un array de objetos Shape. El siguiente código crea un Rectangle y un Ellipse2D.Double y los introduce en el array shapes.

```
shapes[0] = new Rectangle (0, 0, 100, 100);

shapes[1] = new Ellipse2D.Double(0.0, 0.0, 100.0,   100.0);
```

Para crear un objeto Shape desde una cadena de texto, primero debemos crear un objeto TextLayout desde el texto de la cadena.

```
TextLayout textTl = new TextLayout("Text",

                    new Font("Helvetica", 1, 96),
```

```
                    new FontRenderContext(null, false,
                                          false)
                    );
```

Las siguientes líneas transforman el TextLayout para que sea centrado en el origen y luego introduce el objeto Shape resultante de la llamada a getOutline dentro del array shapes.

```
AffineTransform textAt = new AffineTransform();

textAt.translate(0,(float)textTl.getBounds().getHeight());

shapes[2] = textTl.getOutline(textAt);
```

Podemos elegir un primitivo accediendo al índice apropiado dentro del array shapes.

```
Shape shape  = shapes[
                      Transform.primitive. getSelectedIndex()
                ];
```

Cómo se realiza el dibujo dependen de las opciones elegidas.

Cuando el usuario elige stroke, se llama a Graphics2D.draw para realizar el dibujo, si se elige text como primitivo, las líneas son recuperadas y el dibujo se hace con el método draw.

Cuando el usuario elige fill, se llama a Graphics2D.fill o Graphics2D.drawString para realizar el dibujado. Cuando el usuario elige stroke and fill, se llama a fill o drawString para rellenar el Shape, y luego se llama a draw para dibujar la línea exterior.

Los tres estilos de línea usados en este ejemplo ancho, estrecho y punteado son ejemplares de BasicStroke.

```
case 0 : g2.setStroke(new BasicStroke(3.0f));
         break;
case 1 : g2.setStroke(new BasicStroke(8.0f));
          break;
case 2 : float dash[] = {10.0f};
         g2.setStroke(new BasicStroke(3.0f,
                          BasicStroke.CAP_BUTT,
                          BasicStroke.JOIN_MITER,
          10.0f, dash, 0.0f));
break;
```

El estilo de punteado de este ejemplo tiene 10 unidades de punteado alternados con 10 unidades de espacio. El principio del patrón del punteado se aplica al principio de la línea la fase de punteado es 0.0.

En este ejemplo se usan tres estilos de dibujo  sólido, gradiente y polka. El dibujo de color sólido es un ejemplar de Color, el gradiente un ejemplar de GradientPaint, y el patrón un ejemplar de TexturePaint.

```
case 0 : g2.setPaint(Color.blue); break;

case 1 : g2.setPaint(new GradientPaint(0, 0,

                                        Color.lightGray,

                                        w-250,h,Color.blue,

                                        false));

break;

case 2 : BufferedImage bi = new BufferedImage(5, 5,

BufferedImage.TYPE_INT_RGB);


         Graphics2D big = bi.createGraphics ();

         big.setColor(Color.blue);

         big.fillRect(0, 0, 5, 5);

         big.setColor(Color.lightGray);

         big.fillOval(0, 0, 5, 5);

         Rectangle r = new Rectangle(0,0,5,5);

         g2.setPaint(new TexturePaint(bi, r));

 break;
```



Texto Tomado de: http://www.programacioncastellano.com

Título Tutorial: Programación Avanzada con Java2D.

Año Publicación: 05-01-2008

## 4.7. JavaBeans.-

En la industria electrónica como en otras industrias se está acostumbrado a utilizar componentes para construir placas, tarjetas, etc. En el campo del software la idea es la misma. Se puede crear una interfaz de usuario en un programa Java en base a componentes: paneles, botones, etiquetas, caja de listas, barras de desplazamiento, diálogos, menús, etc.

Si se ha utilizado Delphi o Visual Basic, ya estamos familiarizados con la idea de componente, aunque el lenguaje de programación sea diferente. Existen componentes que van desde los más simples como un botón hasta otros mucho más complejos como un calendario, una hoja de cálculo, etc.

Los primeros componentes que tuvieron gran éxito fueron los VBX (Visual Basic Extension), seguidos a continuación por los componentes OCX (OLE Custom Controls). Ahora bien, la principal ventaja de los JavaBeans es que son independientes de la plataforma.

Muchos componentes son visibles cuando se corre la aplicación, pero no tienen por qué serlo, solamente tienen que ser visibles en el momento de diseño, para que puedan ser manipulados por el Entorno de Desarrollo de Aplicaciones (IDE).

Podemos crear una aplicación en un IDE seleccionando los componentes visibles e invisibles en una paleta de herramientas y situarlas sobre un panel o una ventana. Con el ratón unimos los sucesos (events) que genera un objeto (fuente), con los objetos (listeners) interesados en responder a las acciones sobre dicho objeto. Por ejemplo, al mover el dedo en una barra de desplazamiento (fuente de sucesos) con el ratón, se cambia el texto (el



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Tecnología JavaBeans.

Año Publicación: 22-04-2007

número que indica la posición del dedo) en un control de edición (objeto interesado en los sucesos generados por la barra de desplazamiento).

Un JavaBean o bean es un componente hecho en software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.

Para ello, se define un interfaz para el momento del diseño (design time) que permite a la herramienta de programación o IDE, interrogar (query) al componente y conocer las propiedades (properties) que define y los tipos de sucesos (events) que puede generar en respuesta a diversas acciones.

Aunque los beans individuales pueden variar ampliamente en funcionalidad desde los más simples a los más complejos, todos ellos comparten las siguientes características:

**Introspection:** Permite analizar a la herramienta de programación o IDE como trabaja el bean

**Customization:** El programador puede alterar la apariencia y la conducta del bean.

**Events:** Informa al IDE de los sucesos que puede generar en respuesta a las acciones del usuario o del sistema, y también los sucesos que puede manejar.

**Properties:** Permite cambiar los valores de las propiedades del bean para personalizarlo (customization).

**Persistence:** Se puede guardar el estado de los beans que han sido personalizados por el programador, cambiando los valores de sus propiedades.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Tecnología JavaBeans.

Año Publicación: 22-04-2007

En general, un bean es una clase que obedece ciertas reglas:

- Un bean tiene que tener un constructor por defecto (sin argumentos)
- Un bean tiene que tener persistencia, es decir, implementar el interface ***Serializable***.

Un bean tiene que tener introspección (instrospection). Los IDE reconocen ciertas pautas de diseño, nombres de las funciones miembros o métodos y definiciones de las clases, que permiten a la herramienta de programación mirar dentro del bean y conocer sus propiedades y su conducta.

**Propiedades**

Una propiedad es un atributo del JavaBean que afecta a su apariencia o a su conducta. Por ejemplo, un botón puede tener las siguientes propiedades: el tamaño, la posición, el título, el color de fondo, el color del texto, si está o no habilitado, etc.

Las propiedades de un bean pueden examinarse y modificarse mediante métodos o funciones miembro, que acceden a dicha propiedad, y pueden ser de dos tipos:

**getter method:** Lee el valor de la propiedad

**setter method:** Cambia el valor de la propiedad.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Tecnología JavaBeans.

Año Publicación: 22-04-2007

Un IDE que cumpla con las especificaciones de los JavaBeans sabe como analizar un bean y conocer sus propiedades. Además, crea una representación visual para cada uno de los tipos de propiedades, denominada editor de propiedades, para que el programador pueda modificarlas fácilmente en el momento del diseño.

Cuando un programador, coge un bean de la paleta de componentes y lo deposita en un panel, el IDE muestra el bean sobre el panel. Cuando seleccionamos el bean aparece una hoja de propiedades, que es una lista de las propiedades del bean, con sus editores asociados para cada una de ellas.

El IDE llama a los métodos o funciones miembro que empiezan por **get**, para mostrar en los editores los valores de las propiedades. Si el programador cambia el valor de una propiedad se llama a un método cuyo nombre empieza por **set**, para actualizar el valor de dicha propiedad y que puede o no afectar al aspecto visual del bean en el momento del diseño.

Las especificaciones JavaBeans definen un conjunto de convenciones (design patterns) que el IDE usa para inferir qué métodos corresponden a propiedades.

```
public void setNombrePropiedad(TipoPropiedad valor)
public TipoPropiedad getNombrePropiedad( )
```

Cuando el IDE carga un bean, usa el mecanismo denominado ***reflection*** para examinar todos los métodos, fijándose en aquellos que empiezan por **set** y **get**. El IDE añade las propiedades que encuentra a la hoja de propiedades para que el programador personalice el bean.

### *Propiedades simples*

Una propiedad simple representa un único valor. Ejemplo

```
private String nombre;

public void setNombre(String nuevoNombre){
    nombre=nuevoNombre;
}
public String getNombre(){
    return nombre;
}
```

En el caso de que dicha propiedad sea booleana se escribe

```
private boolean conectado=false;
//métodos set y get de la propiedad denominada
Conectado
public void setConectado(boolean nuevoValor){
    conectado=nuevoValor;
}
public boolean isConectado(){
    return conectado;
}
```

### *Propiedades indexadas.-*

*Una propiedad indexada representa un array de valores.*

```
private int[] numeros={1,2,3,4};

public void setNumeros(int[] nuevoValor){
    numeros=nuevoValor;
}
public int[] getNumeros(){
    return numeros;
}
```

### *Propiedades ligadas (bound).-*

Los objetos de una clase que tiene una propiedad ligada notifican a otros objetos (listeners) interesados, cuando el valor de dicha propiedad cambia, permitiendo a estos objetos realizar alguna acción. Cuando la propiedad cambia, se crea un objeto (event) que contiene información acerca de la propiedad (su nombre, el valor previo y el nuevo valor), y lo pasa a los otros objetos (listeners) interesados en el cambio.

### *Propiedades restringidas (constrained)*

Una propiedad restringida es similar a una propiedad ligada salvo que los objetos (listeners) a los que se les notifica el cambio del valor de la propiedad tienen la opción de vetar (veto) cualquier cambio en el valor de dicha propiedad.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Tecnología JavaBeans.

Año Publicación: 22-04-2007

## 4.8 JDOM.-

JDOM es un API para leer, crear y manipular documentos XML de una manera sencilla y muy intuitiva para cualquier programador en Java, en contra de otras APIS tales como DOM y SAX, las cuales se idearon sin pensar en ningún lenguaje en concreto, de ahí que resulte un poco incomodo su utilización.

En este momento exploraremos un poco el API, crearemos una pequeña aplicación que lea un documento XML y sacaremos de él aquellas partes que nos interese. Para abrir boca, imagínate este trozo de XML:

**...<site>javahispano</site>...**

Se imagina acceder al texto del elemento site de esta manera:

```
String nombre = site.getText ();
```

**Donde se encuentra JDOM.-** Para comenzar haremos una visión de pájaro para ver donde encajamos JDOM en un pequeño esquema.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: JDOM.

Año Publicación: 13-09-2008

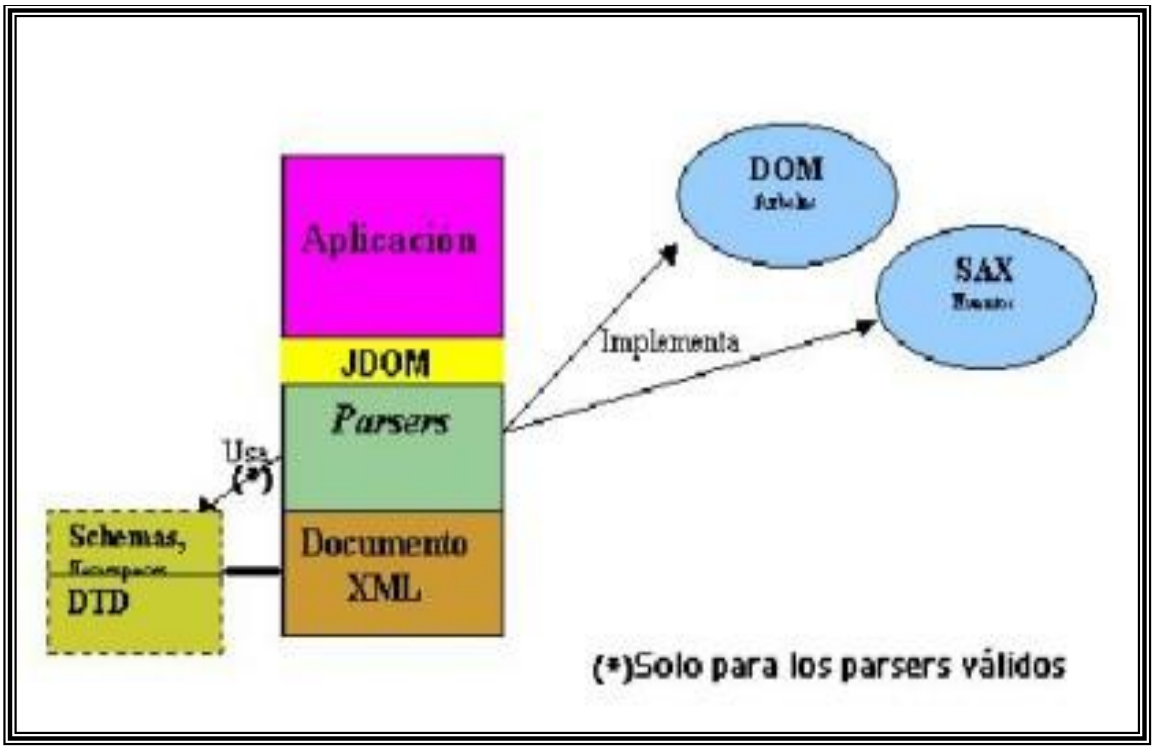



**Figura 22.- Esquema JDOM.**

Antes de seguir adelante permitirme una aclaración: DOM y SAX son dos especificaciones que como tal no podemos trabajar con ellas, pero si con las implementaciones de dichas especificaciones, es decir, los parsers: Xerces, XML4j, Crimson, Oracle's parsers etc.

Pues bien un último apunte, la API JDOM no es un parser, de hecho, usa un parser para su trabajo, JDOM "solo" nos aporta una capa de abstracción en el tratado de documentos XML facilitándonos bastante la tarea como veremos enseguida, de hecho no tendremos que ser unos gurús de DOM y SAX para poder trabajar con XML desde Java.

Como JDOM fue diseñado usando List y Map de la API Java 2 Collections, tendremos que utilizar el jdk1.2.0 y versiones posteriores.

**Estructura de JDOM**



Texto Tomado de: http://www.wikipedia.com

Documento HTML: JDOM.

Año Publicación: 13-09-2008

El API está formado por 5 packages. De entre ellas comentamos lo siguiente que será más que suficiente para utilizar el API.

- El package org.jdom destacamos las clases: Document que representará el documento XML, Element que representará el elemento o etiqueta que forma el documento, y la clase Attribute que como bien imaginaras representa los atributos que puedan tener los elementos.
- El package org.jdom.adapters albergará todas las clases adaptadoras (ver patrón de diseño Adapter, Thinking in patterns) ya que no todos los parsers DOM tienen la misma API. Más tarde quedará más claro su función.
- El package org.jdom.input albergara las clases builder para construir los documentos XML.
- El package org.jdom.output albergara las clases que utilizaremos para dar salida a nuestra clase Document.

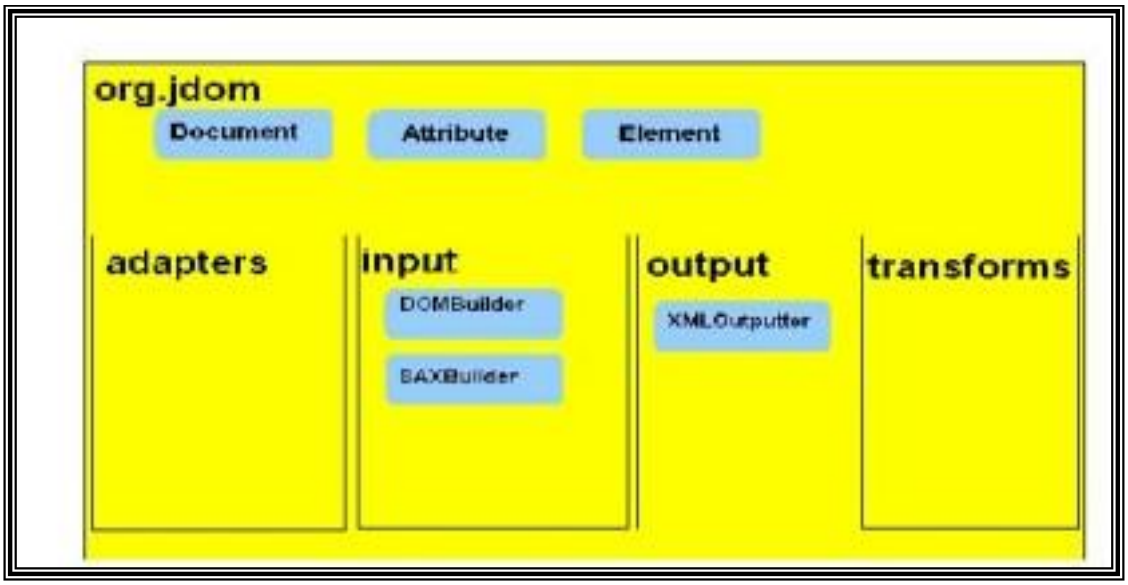



**Figura 23.- Estructura de JDOM.**

**Un poco de teoría**

Como dice JDOM usaba los parsers para hacer su trabajo, pues bien, para decirle a JDOM que parser utilizar utilizaremos uno de los siguientes métodos:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: JDOM.

Año Publicación: 13-09-2008

**public SAXBuilder (String parserClass, boolean validation)**

El primer parámetro es el parser que vamos a utilizar, por defecto se utilizará el parser Xerces.

El segundo parámetro es para decirle si queremos que el parser cumpla sus obligaciones de validación.

**public DOMBuilder (String adapterClass, boolean validation)**

El primer parámetro es la clase adaptadora que vamos a utilizar para el parser que utilizaremos.

El segundo parámetro es igual que el del SAXBuilder.

Ahora al builder le daremos la orden de parsear el documento XML con el método build (), cuya forma es:

**Document build (File file)**

Muy bien ya tenemos el documento almacenado en la clase Document. Finalmente vamos a aprender unos cuantos métodos más para recuperar la información que deseemos:

**Element getRootElement ():** Coger el nodo raíz del documento.

Estos métodos pertenecen a la clase Element:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: JDOM.

Año Publicación: 13-09-2008

| | |
|---|---|
| **String getText ()** | Capturar el texto de una etiqueta o elemento. |
| **List getChildren ()** | Coger todos los elementos que cuelgan del Element. |
| **List getChildren (String nombre)** | Coger todos los elementos que tengan ese nombre. |
| **List getMixedContent ()** | Para recuperar todo (comentario, elemento) |
| **Element getChild (String nombre)** | Coger el primer hijo que tenga ese nombre. |
| **String getAttributeValue (String nombre)** | Coger el valor del atributo que pasamos como parámetro |
| **Attribute getAttribute (String nombre)** | Coger el atributo que tenga ese nombre. |
| **Attribute: String getValue ()** | Para recuperar el valor de ese atributo. |



Texto Tomado de: http://www.wikipedia.com

Documento HTML: JDOM.

Año Publicación: 13-09-2008

## 4.9 Apache Ant.-

Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es similar a Make pero sin las engorrosas dependencias del sistema operativo.

Esta herramienta, hecha en Java, tiene la ventaja de no depender de las órdenes de shell de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma.

### 4.9.1.- Fichero “build.xml”.-

Debajo se muestra un archivo de ejemplo (build.xml) para una aplicación "Hola mundo" en Java. El archivo define tres objetivos - clean, compile y jar, cada uno de los cuales tiene una descripción asociada. El objetivo *jar* lista el objetivo *compile* como dependencia. Esto le dice a ANT que, antes de empezar el objetivo *jar*, debe completar el objetivo *compile*.

Dentro de cada **objetivo están las acciones** que debe tomar ANT para construir el objetivo. Por ejemplo, para construir el objetivo *compile* ANT debe primero crear un directorio llamado classes (ANT sólo lo hará si éste no existe previamente) y luego llamar al compilador de Java.

A continuación detallamos un ejemplo sencillo de un archivo de tareas ANT, en un archivo build.xml:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="borrar archivos temporales">
        <delete dir="classes" />
    </target>
    <target name="compile"
     description="compilar el código java a un archivo class">
        <mkdir dir="classes" />
        <javac srcdir="." destdir="classes" />
    </target>
    <target name="jar" depends="compile"
     description="crear un archivo Jar para la aplicación">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class" />
            <manifest>
                <attribute name="Main-Class" value="HelloProgram" />
            </manifest>
        </jar>
    </target>
</project>
```

**Figura 24.- Tarea ANT con JDOM.**

### 4.9.2.- Portabilidad.-

Una de las primeras ayudas de Ant fue solucionar los problemas de portabilidad de Make. En un Makefile las acciones necesarias para crear un objetivo se especifican como órdenes de Intérprete de comandos que son específicos de la plataforma, normalmente un shell de Unix. Ant resuelve este problema proveyendo una gran cantidad de funcionalidades por él mismo, que pueden garantizar que permanecerán (casi) idénticas en todas las plataformas.

Por ejemplo, en el ejemplo build.xml debajo del objetivo clean borra el directorio classes y todo su contenido. En un Makefile esto normalmente se haría con la orden:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

**rm -rf classes/**

"rm" es una orden específica de Unix que probablemente no estará disponible si el Makefile se usa en un entorno no Unix como Microsoft Windows. En un archivo de construcción Ant se puede conseguir lo mismo usando una orden propia:

**<delete dir="classes" />**

Una discrepancia común entre diferentes plataformas es la manera en que las rutas de directorios se especifican. Unix usa la barra (/) para delimitar los componentes de una ruta, mientras que Windows usa la barra invertida (\). Los archivos de Ant permiten al autor elegir su convención favorita, barras o barras invertidas para directorios, comas o punto y coma para separar rutas. Ant lo convierte todo al formato apropiado para la plataforma actual.

### 4.9.3.- Historia.-

ANT fue creado por James Duncan Davidson mientras realizaba la transformación de un proyecto de Sun Microsystems en Open Source (concretamente la implementación de Servlets y JSP de Sun que luego se llamaría Jakarta Tomcat). En un entorno cerrado Make funcionaba correctamente bajo plataforma Solaris, pero para el entorno de open source, donde no era posible determinar la plataforma bajo la que se iba a compilar, era necesaria otra forma de trabajar. Así nació Ant como un simple intérprete que cogía un archivo XML para compilar Tomcat independientemente de la plataforma sobre la que operaba. A partir de este punto la herramienta fué adoptando nuevas funcionalidades y actualmente es un estándar en el mundo Java.

### 4.9.4.- Limitaciones.-

- Al ser una herramienta basada en XML, los archivos Ant deben ser escritos en XML. Esto es no sólo una barrera para los nuevos usuarios, sino también un problema en los proyectos muy grandes, cuando se construyen archivos muy grandes y complejos. Esto quizá sea un problema común a todos los lenguajes XML, pero la granularidad de las tareas de Ant significa que los problemas de escalabilidad llegan pronto. Si te surgen estos problemas, probablemente tengas mal estructurado el proyecto.

- La mayoría de las antiguas herramientas — las que se usan todos los días, como <javac>, <exec> y <java> tienen malas configuraciones por defecto, valores para opciones que no son coherentes con las tareas más recientes. Ésta es la maldición de la compatibilidad hacia atrás: cambiar estos valores supone estropear las herramientas existentes.

- Cuando se expanden las propiedades en una cadena o un elemento de texto, las propiedades no definidas no son planteadas como error, sino que se dejan como una referencia sin expandir (ej.: ${unassigned.property}). De nuevo, ésta es una cuestión de la compatibilidad hacia atrás, incluso se reconoce que tener la herramienta desactivada es normalmente la mejor opción, al menos hasta el punto que el mítico producto "Ant2.0" falle en propiedades no asignadas.

- No es un lenguaje para un flujo de trabajo general, y no debería ser usado como tal. En particular, tiene reglas de manejo de errores limitadas, y no tiene persistencia de estado, así que no puede ser usado con confianza para manejar una construcción de varios días.

### 4.9.5.- *Apache Ant, creando un archivo build.xml*.-

Uno de los inconvenientes en la construcción de software con más de dos ficheros es el proceso de compilación: orden de compilación de ficheros, dependencias, rutas de librerías internas, etc. En esta entrada vamos a dar los



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

primeros pasos en una de las soluciones más comunes para proyectos Java: Apache Ant.

En primer lugar creamos un fichero build.xml en la raíz de nuestro proyecto y definimos su nombre:

**<project name="Proyecto"> </project>**

Ant, al igual que otras herramientas de construcción, se basa en el concepto de objetivos o ***targets*** cuya definición engloba tanto las dependencias previas como los pasos a seguir para conseguirlo.

Vamos a comenzar definiendo un objetivo de preparación llamado ***init*** que será el encargado de crear un directorio ***classes*** donde guardaremos los ficheros ***.class*** resultantes de la compilación y el directorio ***build*** para el ***.jar*** final. Para ello basta incluir dentro de <project> las siguientes líneas:

```
<target name="init">
        <mkdir dir="classes" />
        <mkdir dir="build" />
</target>
```

**Figura 25.- Target init.**

Como podemos ver los objetivos se delimitan con etiquetas **<target>** y un nombre. Dentro de ellos se enumeran los pasos que se han de seguir para alcanzar el objetivo, en este caso ha de crear directorios.

Si queremos alcanzar el objetivo ***init*** basta con realizar:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
$ ant init
Buildfile: build.xml

init:
    [mkdir] Created dir: /home/chernando/proyecto/classes
    [mkdir] Created dir: /home/chernando/proyecto/build

 BUILD SUCCESSFUL
Total time: 0 seconds
```

**Figura 26.- Ejecución de la Tarea init con ANT.**

Es hora de compilar nuestro proyecto, vamos a definir el objetivo ***compile***. Ahora bien, la compilación depende de la creación del directorio ***classes*** que se realiza objetivo anterior. Con esto en cuenta basta con incluir:

```
<target name="compile" depends="init">
        <javac srcdir="src" destdir="classes" />
</target>
```

**Figura 27.- Tarea compile con ANT.**

La dependencia se fija en la declaración del ***target*** de tal manera que se garantiza su cumplimiento antes de comenzarla. Nuestro código está en el directorio ***src*** y el resultado de la compilación se lleva al directorio ***classes***.

Importante notar que esta vez estamos usando **<javac>** esto es lo que Ant se denomina tarea. Hay muchas tareas predefinidas, consultad el manual de ant. Con nuestro proyecto compilado vamos a generar el ***.jar*** que distribuiremos haciendo uso de un nuevo objetivo llamado ***build***.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
<target name="build" depends="compile">
        <jar destfile="build/proyecto.jar" basedir="classes" />
</target>
```

**Figura 28.- Tarea build con ANT.**

En este caso dependemos de los frutos de ***compile*** y utilizamos la tarea ***jar*** que se encarga de empaquetar todo el contenido del directorio ***classes*** en el fichero ***proyecto.jar***.

Finalmente incluiremos un nuevo objetivo para limpiar todo el entorno, el objetivo ***clean***:

```
<target name="clean">
        <delete dir="classes" />
        <delete dir="build" />
</target>
```

**Figura 29.- Tarea clean con ANT.**

A estas alturas es fácil entender que lo único que realiza es eliminar los directorios de trabajo dejando el entorno limpio del proceso de compilación. Resumiendo nuestro fichero build.xml es:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
<project name="Proyecto">
        <target name="init">
                <mkdir dir="classes" />
                <mkdir dir="build" />
        </target>
        <target name="compile" depends="init">
                <javac srcdir="src" destdir="classes" />
        </target>
        <target name="build" depends="compile">
                <jar destfile="build/proyecto.jar" basedir="classes" />
        </target>
        <target name="clean">
                <delete dir="classes" />
                <delete dir="build" />
        </target>
</project>
```

**Figura 30.- Ejemplo completo de un archivo build.xml.**

### 4.9.6.- Estructura de un Fichero (build.xml).-

Los Ficheros-de-construcción están escritos en XML.

Esta es la estructura básica de un Fichero-de-construcción llamado build.xml

```
<project
      name="nombreProyecto"
      default="nombreObjetivo"
      basedir="."
>
   <target name="nombreObjetivo">
     <property file="localizaciónFichPropiedad"/>
```



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
  </target>
  <target name="nombreObjetivo">
    <tarea1/>
    <tarea2/>
  </target>
</project>
```

**Figura 31.- Estructura Básica de un build.xml.**

- **<project>**
  - Estos ficheros tienen un elemento raíz que es un Proyecto (project)
    - Un Proyecto contiene tres atributos
      - **name**
        - Es el nombre del proyecto
      - **default**
        - El Objetivo (target) por defecto a utilizar cuando no se ha suministrado ningún Objetivo en el comando ant que ejecuta el Fichero-de-construcción build.xml
      - **basedir**
        - Es el directorio base desde donde todos los cálculos de path se van a realizar
    - Un Proyecto contiene al menos un Objetivo (target)
      - Estos Objetivos contienen a su vez elementos de **Tareas (tasks)**
        - Un Objetivo es un conjunto de Tareas que queremos que se ejecuten
          - Una de las Tareas más comunes son javac para compilar código Java y java para ejecutar una Clase Java
    - Cualquier proyecto se basa en las Propiedades (properties)
      - Las Propiedades son ficheros que contienen nombres de propiedades y sus valores correspondientes

Las propiedades tienen diferentes combinaciones de atributos

```
<property file="localizaciónFichPropiedad"/>
```

- Utilizamos el atributo file de una propiedad para indicar la localización de un fichero de propiedades.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
<property name="nombrePropiedad" location="localizaciónFichPropiedad" />
```

- Si se utiliza el atributo name entonces la localización de la Propiedad se realiza a través del atributo location

```
<property name="nombrePropiedad" value="valorPropiedad" />
```

- Dentro del Fichero-de-Construcción podemos declarar e inicializar una Propiedad.
  - El atributo value de una Propiedad puede tener
    - El valor de esa propiedad
    - La localización de un fichero de propiedades que contenga varias propiedades con sus valores

Los Objetivos tienen diferentes atributos

```
<target name="nombreObjetivo">
  ...
</target>
```

- El atributo name indica el nombre del Objetivo.
- Este es el único atributo del elemento target que es obligado

```
<target name="nombreObjetivo" depends="nomObjetivo1, nomObjetivo2">
  ...
</target>
```

- Un objetivo puede depender de cero o más Objetivos.
  - Podríamos tener el objetivo de por ejemplo compilar y otro objetivo que fuera crear un desplegado de nuestra aplicación.
    - Debido a que sólo podemos desplegar la aplicación si hemos compilado antes, el Objetivo de crear un desplegado dependerá del Objetivo de compilación.
  - Un Objetivo sólo se ejecuta una vez, incluso cuando más de un Objetivo depende de éste.
  - Para indicar que se depende de algún objetivo, se indica con el atributo depends. Si se dependiera de más de un objetivo, estos los separamos con comas.

```
<target name="nombreObjetivo" if="nombrePropiedad">
```



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
  ...
</target>
```

- Si el nombre de la propiedad tiene un valor, el Objetivo se ejecutará
- Si por el contrario el nombre de la propiedad no tiene ningún valor, el Objetivo no se ejecutará

```
<target name="nombreObjetivo" unless="nombrePropiedad">
  ...
</target>
```

- Si el nombre de la propiedad tiene un valor, el Objetivo no se ejecutará
- Si por el contrario el nombre de la propiedad no tiene ningún valor, el Objetivo se ejecutará

Especificación de una tarea

```
<tarea1 atributo1="valor1" atributo2="valor2"/>
```

- Las tareas
  - Son trozos de código que pueden ser ejecutados.
  - Pueden tener múltiples atributos.
    - El valor de un atributo puede contener referencias hacia una Propiedad
      - Estas referencias serán resueltas antes de que la tarea se haya ejecutado

Para referenciar a un nombre de propiedad desde una Tarea

```
<target name="nombreObjetivo">
    <tarea1 atributo1="${nombrePropiedad}" atributo2="${os.name}"/>
</target>
```

- Si queremos referencia a un nombre de propiedad desde una Tarea lo haremos con la sintaxis mostrada en el atributo1
- También podemos acceder a las Propiedades propias de cada Sistema Operativo, como en el atributo2



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

### 4.9.7.- Ejecución de un Fichero (build.xml).-

Para ejecutar uno de estos ficheros tenemos que tener instalado en nuestro sistema la herramienta Ant de Apache

```
shell> ant
```

- Ejecuta el objetivo del atributo default del elemento <project>

```
shell> ant nombreObjetivo
```

- Ejecuta el objetivo que le acabamos de indicar como argumento del comando ant.

### 4.9.8.- *Tareas nativas para Apache Ant*.-

Buscando una forma de crear accesos directos y de acceder al registro Windows desde Apache Ant he encontrado Orangevolt Ant Tasks.

Se trata de 17 tareas que pueden ser de utilidad en la creación de scripts Ant para instalar programas en Windows y Linux:

- ***win32.properties***: Obtiene las propiedades de Windows (rutas a escritorio, menú de inicio, etc.).
- ***win32.registry***: Lee y actualiza el registro de Windows.
- ***win32.shortcut***: Crea una acceso directo a un fichero.
- ***sfx***: Crea un autoextraíble nativo a partir de un fichero “ZIP”.
- ***jstub***: Crea un ejecutable nativo a partir de un fichero “jar”.
- ***jnlp***: Crea un fichero JNLP para Java Web Start.
- ***preferences***: Permite acceder al API de preferencias de Java.
- ***properties***: Escribe un fichero de propiedades Java.
- ***find***: Busca un fichero en el disco duro a partir de un patrón.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

- ***compare***: Compara propiedades o listas de propiedades independientemente de su tipo.
- ***execute***: Ejecuta un bloque de tareas una o más veces dependiendo de una condición.
- ***call***: Ejecuta una tarea una o más veces dependiendo de un condición.
- ***os.properties***: Obtiene una lista de propiedades con información acerca del SO.
- ***unix.link***: Crea un enlace en Unix.
- ***unix.kde.shortcut***: Crea un acceso directo a un fichero.
- ***unix.kde.directoryshortcut***: Crea un acceso directo a directorio.
- ***HttpGet, HttpPost y Http***Read: Permite cargar y descargar información mediante HTTP.

Algunas de estas tareas son tan específicas del SO que utilizan código nativo (en forma de DLLs, etc.). Sin embargo, desde Ant no tenemos que preocuparnos de su gestión, ya que las tareas se encargan de ello de forma transparente. En el siguiente ejemplo muestra cómo crear un atajo en el escritorio de Windows:



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

```
view plain  copy to clipboard  print  ?
<project name="install" default="create-shortcut">

  <!-- Definimos las tareas de Orangevolt -->
  <taskdef resource="com/orangevolt/tools/ant/taskdefs.properties"/>

  <target name="create-shortcut">

    <!-- Obtiene propiedades de Windows -->
    <win32.properties/>

    <!-- Crear un acceso directo a Eclipse en el escritorio -->
    <win32.shortcut file="${win32.common.desktop}/Eclipse.lnk"
       execute="C:/eclipse/eclipse.exe"
       comment="Acceso directo a Eclipse"/>

  </target>

</project>
```

**Figura 32.- Tarea Ant para crear un Acceso Directo.**



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Apache-Ant.

Año Publicación: 11-07-2008

## 4.10 Netbeans.-


**Figura 33.- Plataforma Netbeans.**

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

### 4.10.1.- Historia.-

NetBeans comenzó como un proyecto estudiantil en República Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad de Charles en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-release en 1997.

Xelfi fue un proyecto divertido para trabajar, ya que las IDEs escritas en Java eran un territorio desconocido en esa época. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. Casi todos ellos siguen trabajando en NetBeans.

Tiempo después, ellos fueron contactados por Roman Stanek, un empresario que ya había estado relacionado con varias iniciativas en la República Checa. Él estaba buscando una buena idea en que invertir, y encontró en Xelfi una buena oportunidad. Ellos se reunieron, y el negocio surgió.



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

El plan original era desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, surgió con la idea de llamarlo NetBeans, con el fin de describir lo que ellos harían. Cuando las especificaciones de los Enterprise JavaBeans salieron, ellos decidieron trabajar con este estándar, ya que no tenía sentido competir con él, sin embargo el nombre de NetBeans se quedó.

En la primavera de 1999, Netbeans DeveloperX2 fue lanzado, soportando Swing. Las mejoras de rendimiento que llegaron con el JDK 1.3, lanzado en otoño de 1999, hicieron a NetBeans una alternativa realmente viable para el desarrollo de herramientas. En el verano de 1999, el equipo trabajó duro para rediseñar a DeveloperX2 en un NetBeans más modular, lo que lo convirtió en la base de NetBeans hoy en día.

Algo más paso en el verano de 1999. Sun Microsystems quería una mejor herramienta de desarrollo de Java, y comenzó a estar interesado en NetBeans. En otoño de 1999, con la nueva generación de NetBeans en Beta, el acuerdo fue realizado.

Sun adquirió otra compañía de herramientas al mismo tiempo, Forté, y decidió renombrar NetBeans a Forté para Java. El nombre de NetBeans desapareció de vista por un tiempo.

Seis meses después, se tomó la decisión de hacer a NetBeans open source. Mientras que Sun había contribuido considerablemente con líneas de código en varios proyectos de código abierto a través de los años, NetBeans se convirtió en el primer proyecto de código abierto patrocinado por ellos. En Junio del 2000 NetBeans.org fue lanzado.

### *4.10.2.- NetBeans en la Actualidad.-*



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

Un proyecto de código abierto no es nada más ni nada menos que un proceso. Toma tiempo encontrar el equilibrio. El primer año, fue crucial como inicio. Los dos años siguientes, se orientó hacia código abierto. Como muestra de lo abierto que era, en los primeros dos años había más debate que implementación.

Con NetBeans 3.5 se mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se re implementó el sistema de ventanas y la hoja de propiedades, y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente. NetBeans IDE 5.0 introdujo un soporte mucho mejor para el desarrollo de nuevos módulos, el nuevo constructor intuitivo de interfaces Matisse, un nuevo y rediseñado soporte de CVS, soporte a Sun ApplicationServer 8.2, Weblogic9 y JBoss 4.

Con Netbeans 6.01 y 6.5 Se dio soporte a frameworks comerciales como son Struts, Hibernate.

### 4.10.3.- La Plataforma NetBeans.-

Durante el desarrollo del NetBeans IDE ocurrió una cosa interesante. La gente empezó a construir aplicaciones usando el NetBeans core runtime con sus propios plug-ins, de hecho, esto se convirtió en un mercado bastante grande.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)

### 4.10.4.- NetBeans IDE.-

El IDE NetBeans es un IDE una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

La versión actual es NetBeans IDE 6.5, la cual fue lanzada el 19 de Noviembre de 2008. NetBeans IDE 6.5 extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS).



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++, mientras el PHP Pack, soporta PHP 5.

**Modularidad.** Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Sun Studio, Sun Java Studio Enterprise, y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans.

Desde Julio de 2006, NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL).



Texto Tomado de: http://www.wikipedia.com

Documento HTML: Plataforma NetBeans.

Año Publicación: 11-02-2009

PLATAFORMA NETBEANS IDE

Swing Evolution 1.1.0 - NetBeans IDE 6.5

Archivo Editar Vista Navegar Fuente Reestructurar Ejecutar Depurar Profile Control de versiones Herramientas Ventana Ayuda

Proyectos | Archivos | Prestaciones

- Paquetes de fuentes
  - resources
  - resources.128x128
  - resources.12x12
  - resources.16x16
  - resources.24x24
  - resources.8X8
  - resources.themes
  - se.persistencia
    - ArchivoJava.java
    - ArchivoXML.java
  - se.proceso.administrarDirectoriosArchivos
  - se.proceso.administrar_plantilla
  - se.proceso.administrar_proyecto
  - se.proceso.componentes
  - se.proceso.generar_archivos_configuracion
  - se.proceso.generar_codigo
  - se.proceso.generar_modulo
  - se.proceso.modificar_apariencia
  - se.proceso.modificar_apariencia.recursos
  - se.vista.SwingEvolutionElegantLayout
  - se.vista.SwingEvolutionElegantLayout.io
  - se.vista.administrar_plantilla

Página de Inicio | ArchivoXML.java

```
7   import java.io.File;
8   import java.io.FileInputStream;
9   import java.io.FileOutputStream;
10  import se.proceso.modificar_apariencia.recursos.VectorFiguraBasica;
11
12  /**
13   * Clase encargado de trabajar de persistencia en xml
14   * @author Darwin Augusto Calle Loja
15   * @author José Daniel Quezada Armijos
16   */
17  public class ArchivoXML {
18
19      /**
20       * Guarda una plantilla en formato xml
21       * @param figuras Vector de figuras a persistir
22       * @param ruta_plantilla Ruta en el disco donde se va ha guardar
23       */
24      public static void guardarPlantilla(VectorFiguraBasica figuras, String ruta_plantilla){
25          try{
26              XMLEncoder escritor=new XMLEncoder(new BufferedOutputStream(new FileOutputStream(ruta_plantilla-
27              escritor.writeObject(figuras);
```

16:4 INS

ArchivoXML - Navegador

Vista de miembros

- ArchivoXML
  - guardarPlantilla(VectorFiguraBasica figuras, String ruta_pl
  - guardarRespaldo(VectorFiguraBasica figuras, String ruta_
  - recuperarObjeto(String ruta_objeto) : VectorFiguraBasica

Salida - Swing_Evolution_1.1.0 (compile-single)

```
init:
deps-jar:
compile-single:
GENERACIÓN CORRECTA (tiempo total: 0 segundos)
```

**Figura 34.- Plataforma Netbeans.**

### 4.10.5.- Instalación de NetBeans.-

A continuación se darán las instrucciones de como instalar NetBeans 5.5 IDE (Standalone Version) en cada una de las plataformas soportadas. Recuerde que el JDK de java ya debe estar instalado en el sistema.

***Instalación en Windows.-***

1. Después de haber bajado el instalador, dar doble clic sobre el icono para iniciar el instalador.
2. Después aparecerá el Acuerdo de Licenciamiento (License Agreement), léalo, acéptelo y de clic en Siguiente (Next).
3. Especificar un directorio vacío donde instalar Netbeans IDE 5.5. Dar clic en Siguiente (Next).
4. Selecciones el JDK que se quiere usar junto con el NetBeans IDE de la lista de selección si hay más de uno. Dar clic en Siguiente (Next).
5. Verifique que el directorio de instalación es el correcto y que se cuenta con espacio suficiente en el sistema para la instalación.
6. De clic en Siguiente (Next) para iniciar la instalación.

***Instalación en Sistemas Operativos Solaris (Solaris OS).-***

1. Navegue hacia el directorio que contiene el instalador.
2. Si es necesario, cambie los permisos del instalador para hacerlo ejecutable escribiendo lo siguiente en la línea de comando (netbeans_installer es el nombre del instalador de NetBeans que se bajó): $ chmod +x netbeans_installer
3. Ejecute el instalador escribiendo lo siguiente en la línea de comando (De nuevo, netbeans_installer es el nombre del instalador de NetBeans que se bajó)): $ ./netbeans_installer
4. Después aparecerá el Acuerdo de Licenciamiento (License Agreement), acéptelo y de clic en Siguiente (Next).
5. Especificar un directorio vacío donde instalar Netbeans IDE 5.5. Dar clic en Siguiente (Next).
6. Selecciones el JDK que se quiere usar junto con el NetBeans IDE de la lista de selección si hay más de uno. Dar clic en Siguiente (Next).

7. Verifique que el directorio de instalación es el correcto y que se cuenta con espacio suficiente en el sistema para la instalación.
8. De clic en Siguiente (Next) para iniciar la instalación.
9. y luego finalizar el instalador

### *Instalación en GNU/Linux*

1. Navegue hacia el directorio que contiene el instalador.
2. Si es necesario, cambie los permisos del instalador para hacerlo ejecutable escribiendo lo siguiente en la línea de comando (netbeans_installer es el nombre del instalador de NetBeans que se bajó): $ chmod +x netbeans_installer
3. Ejecute el instalador escribiendo lo siguiente en la línea de comando (De nuevo, netbeans_installer es el nombre del instalador de NetBeans que se bajó)): $ ./netbeans_installer
4. Después aparecerá el Acuerdo de Licenciamiento (License Agreement), acéptelo y de clic en Siguiente (Next).
5. Especificar un directorio vacío donde instalar Netbeans IDE 5.5. Dar clic en Siguiente (Next).
6. Selecciones el JDK que se quiere usar junto con el NetBeans IDE de la lista de selección si hay más de uno. Dar clic en Siguiente (Next).
7. Verifique que el directorio de instalación es el correcto y que se cuenta con espacio suficiente en el sistema para la instalación.
8. De clic en Siguiente (Next) para iniciar la instalación.

### *Instalación en Macintosh OS X*

1. Bajar el instalador (tarball - .tar.gz) en el directorio deseado,
2. Dar doble clic sobre el icono para desempaquetar el contenido en el sistema.

## Proceso de liberación de versiones de NetBeans

- Se debe lanzar una nueva versión aproximadamente cada seis meses, dos por año, esa es la meta.
- Siempre se necesita estar mirando hacia la nueva versión y la siguiente. No todas las versiones son iguales. Se puede planear muchos cambios en una versión, pero mucho trabajo de estabilización en la siguiente. Eso no significa

que se tolerará una versión con muchos errores, pero como NetBeans evoluciona, es predecible que ocasionalmente ciertas funcionalidades serán rescritas significativamente.

- A pesar que la agenda general dice que deben haber una nueva versión cada seis meses, las fechas exactas deben ser fijadas.
- La agenda de liberación de una nueva versión consiste en:

**1. Congelamiento de características.** Esto se realiza después de un tiempo que no serán introducidas nuevas funcionalidades. Solo serán permitidos en el CVS cambios en el código fuente a causa de corrección de errores.

- Se crea una rama en el CVS para agregar correcciones de errores y cambios en la documentación. La convención para el nombre de esta rama será en el CVS será release donde NN es la versión pendiente a liberar.
- después del congelamiento de características, los cambios en la interfaz de usuario deben ser mínimos. Cualquier cambio pendiente en la interfaz de usuario debe ser comunicado por adelantado y solo debe hacerse para corrección de errores.
- Durante la etapa de estabilización, los binarios son marcados como NB X.Y beta. Construcciones diarias estarán disponibles para bajar y todos los usuarios están invitados a probar el software. Se espera que los desarrolladores respondan rápidamente a los reportes de errores. Los reportes de errores deben ser preferiblemente llenados en la base de datos de errores, pero los desarrolladores deben también monitorear los usuarios de las listas de distribución y responder las recomendaciones aquí escritas.
- Durante la etapa de estabilización, el README y las notas de la versión son escritas. Los primeros borradores de estos dos documentos críticos deben preferiblemente hacerlos antes que después.

**2. Etapa de estabilización.** En la cual los errores son localizados y arreglados.

**3. Release candidate 1.** Si ningún problema serio es encontrado la última versión candidate a ser liberada se convierte en la versión final. Debe haber por lo menos una semana entre la última versión candidata y la versión final.

- después que la primera versión candidate es hecha, todos los cambios en el código fuente deben ser negociados por adelantado escribiendo un post en la lista de distribución de nbdev@netbeans.org nbdev 40 netbeans.org.

**4. Release candidate 2**.- La segunda versión candidate.

**5. Final release.-** La final, versión publica de la nueva versión de NetBeans.

**4.10.6.- Netbeans en el Desarrollo de Interfaces Gráficas.-**

La plataforma Netbeans actualmente contribuye de manera rápida y eficaz a la hora de realizar la creación de Interfaces Gráficas de Usuario (GUI) a los programadores de la plataforma JAVA, es así que gracias a la modularidad de los componentes JavaBeans existentes dentro de esta herramienta la creación de Interfaces Gráficas se hace sumamente fácil.

Debido a la tecnología JavaBeans esta aplicación se la ha considerado como una herramienta capaz de reconocer componentes de software visuales que contengan las especificaciones de la tecnología, y con ello poder lograr la utilización de estos beans dentro de la herramienta como un componente de software más.

Los componentes de software visuales para la creación de GUI dentro de la plataforma se los denomina Controles Swing, estos controles son parte del estándar J2SE que incluye muchas características gráficas y además APIs para la personalización avanzada de estos.

Dentro de la plataforma Netbeans existen los siguientes controles visuales:

- Botones.-
- Cajas de Texto.-
- Paneles de Contenido.-
- Botones de Selección.-
- Listas.
- Árboles.
- Listas Desplegables.
- Controles de Distribución, etc.

A continuación se presenta la plataforma Netbeans en la creación de componentes Visuales (GUI):

# Creación de Interfaces Graficas de Usuario (GUI)

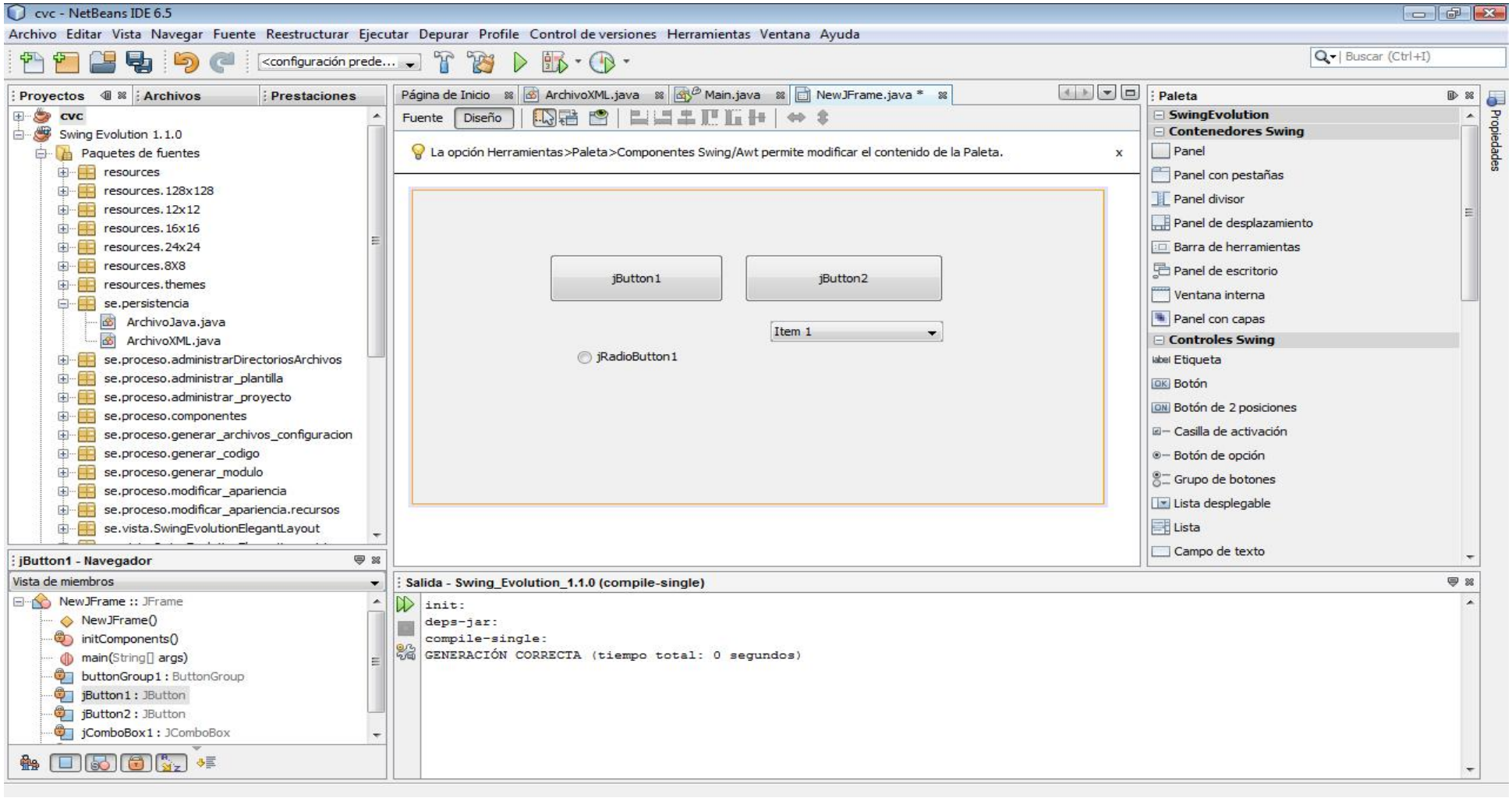
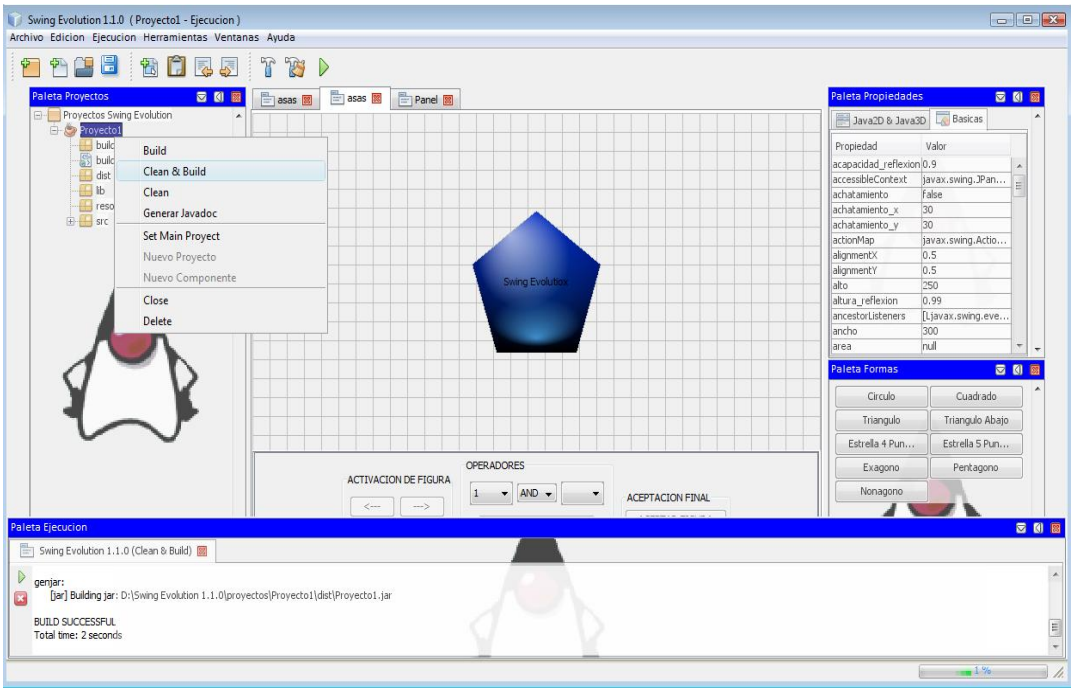


**Figura 35.- Componentes Visuales “Controles Swing”.**

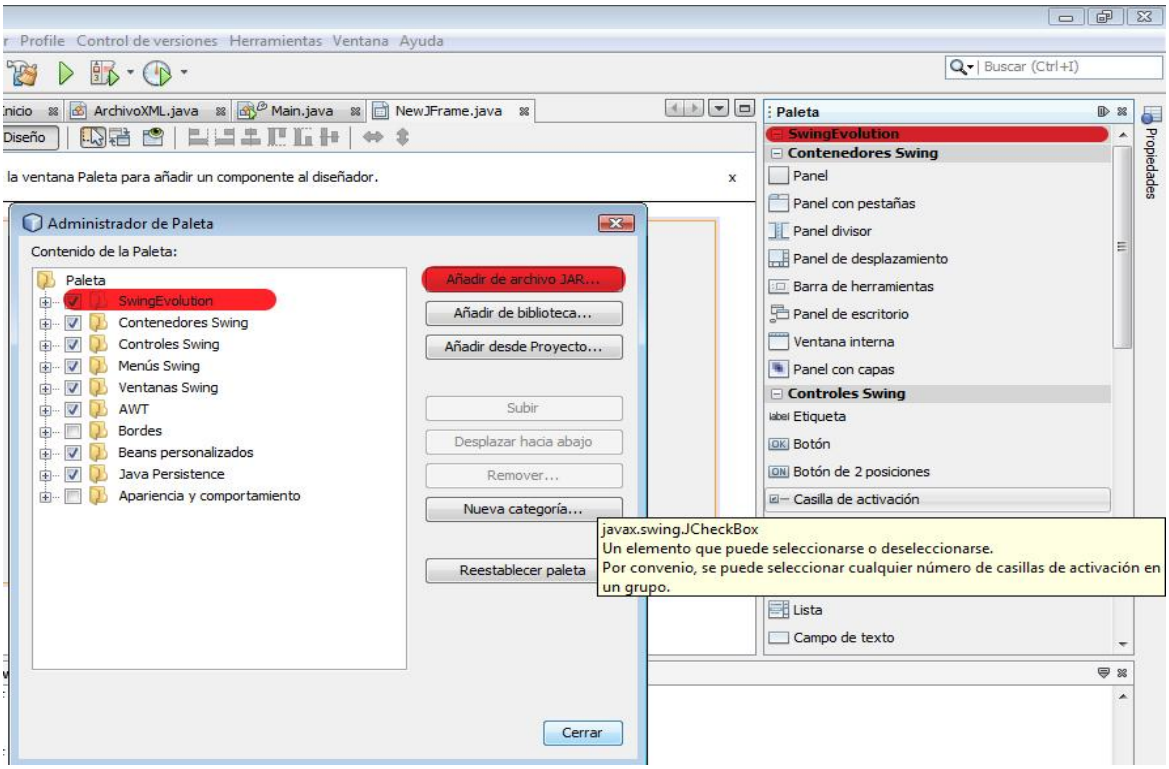
### 4.10.7.- Acoplamiento de un Componente Visual (Swing Evolution) en la plataforma Netbeans.-

Para el acoplamiento de un módulo JAR generado por la aplicación **Swing Evolution** se realizan los siguientes pasos:

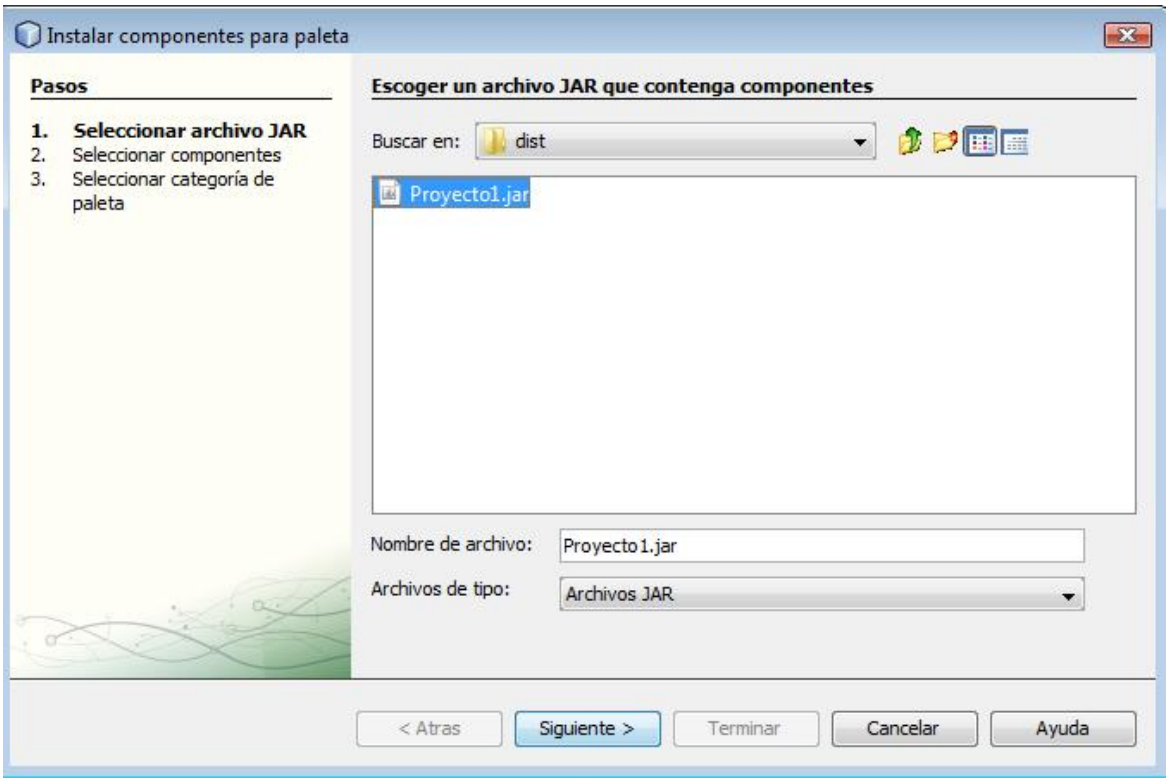
1. Crear un proyecto en la Aplicación Swing Evolution, personalizar los componentes según las necesidades del usuario, una vez terminada la personalización de los controles generar el **Modulo .JAR del proyecto,** haciendo click en la opción **"Clean&Build".**



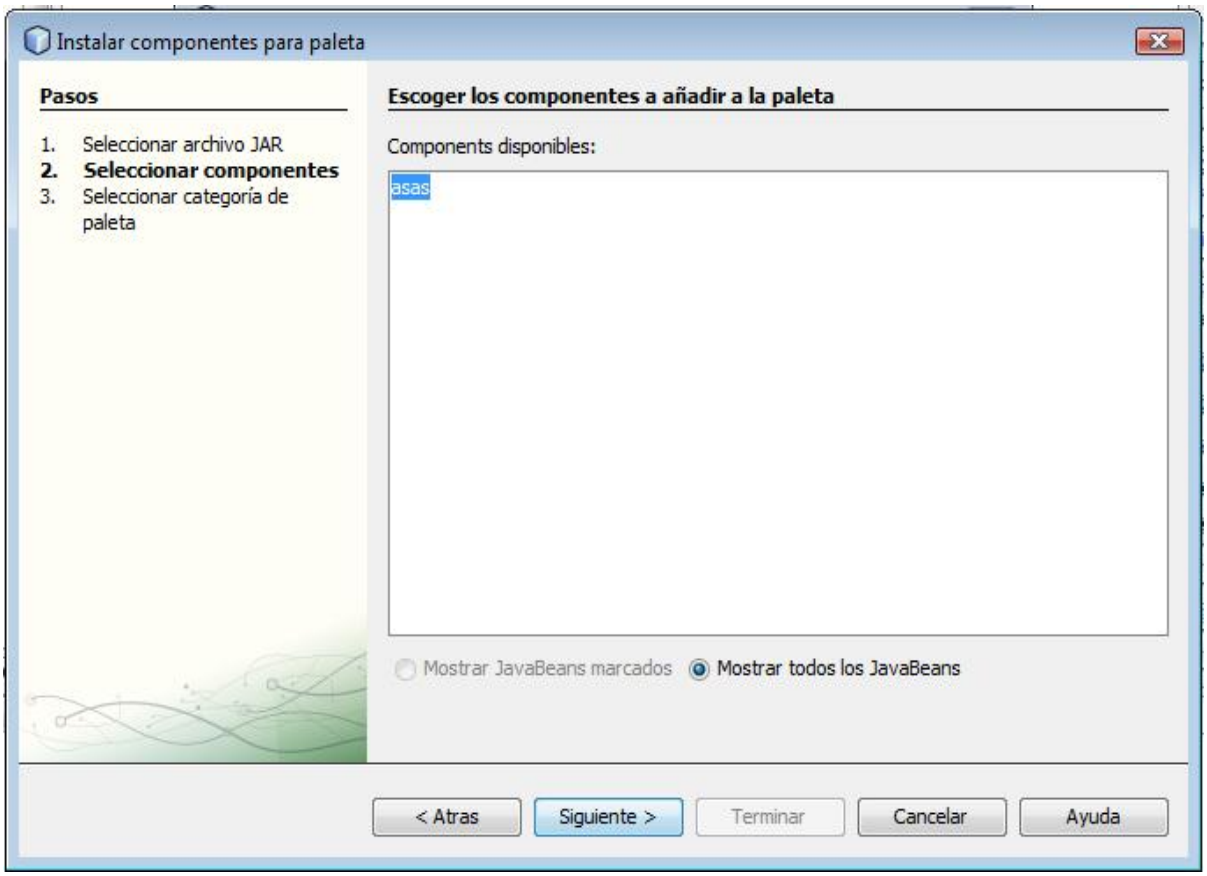

2. En la plataforma Netbeans crear una categoría dentro de la Paleta de Controles Swing con cualquier Nombre, en nuestro caso se llamará **"Swing Evolution",** hacemos click derecho en la paleta y seleccionamos **Administrador de Paleta,** seleccionamos la opción **Añadir de Archivo Jar,** buscamos el archivo **.JAR** dentro de la carpeta de proyectos de la aplicación **"…/Proyecto1/dist/Proyecto.jar"** y finalmente se agrega el componente a la paleta de controles Swing.



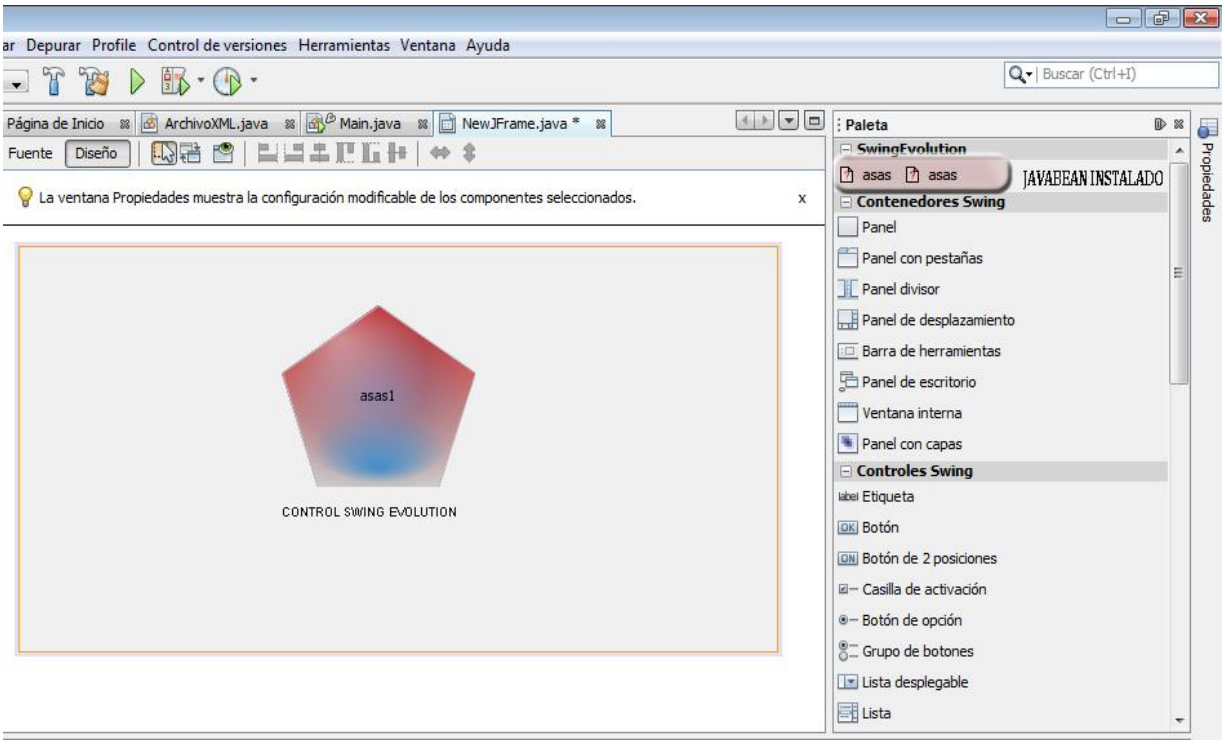

3. Seleccionamos el Modulo JavaBean .JAR del proyecto Swing Evolution.




4. Seleccionamos el Archivo JavaBean que reconoce Netbeans.




5. Y finalizamos el Wizard y se realiza el acoplamiento del Control Visual Swing Evolution en Netbeans como se muestra en la figura.


ar Depurar Profile Control de versiones Herramientas Ventana Ayuda
Buscar (Ctrl+I)
Página de Inicio
ArchivoXML.java
Main.java
NewJFrame.java *
Fuente
Diseño
La ventana Propiedades muestra la configuración modificable de los componentes seleccionados.
asas1
CONTROL SWING EVOLUTION
Paleta
SwingEvolution
asas
asas
JAVABEAN INSTALADO
Contenedores Swing
Panel
Panel con pestañas
Panel divisor
Panel de desplazamiento
Barra de herramientas
Panel de escritorio
Ventana interna
Panel con capas
Controles Swing
Etiqueta
Botón
Botón de 2 posiciones
Casilla de activación
Botón de opción
Grupo de botones
Lista desplegable
Lista
Propiedades

## 4.11 Archivos en Java.-

Una aplicación Java puede escribir en un archivo, salvo que se haya restringido su acceso al disco mediante políticas de seguridad. La dificultad de este tipo de operaciones está en que los sistemas de ficheros son distintos en cada sistema y aunque Java intentar aislar la configuración específica de un sistema, no consigue evitarlo del todo.

### 4.11.1.- Clase File.-

En el paquete **java.io** se encuentra la clase **File** pensada para poder realizar operaciones de información sobre archivos. No proporciona métodos de acceso a los archivos, sino operaciones a nivel de sistema de archivos (listado de archivos, crear carpetas, borrar ficheros, cambiar nombre,...).construcción de objetos de archivo.

Utiliza como único argumento una cadena que representa una ruta en el sistema de archivo. También puede recibir, opcionalmente, un segundo parámetro con una ruta segunda que se define a partir de la posición de la primera.

```
File archivo1=new File("/datos/bd.txt");

File carpeta=new File ("datos");
```

El primer formato utiliza una ruta absoluta y el segundo una ruta relativa. La ruta absoluta se realiza desde la raíz de la unidad de disco en la que se está trabajando y la relativa cuenta desde la carpeta actual de trabajo.

Otra posibilidad de construcción es utilizar como primer parámetro un objeto File ya hecho. A esto se añade un segundo parámetro que es una ruta que cuenta desde la posición actual.

```
File carpeta1=new File("c:/datos");

File archivo1=new File(carpeta1,"bd.txt");
```

Si el archivo o carpeta que se intenta examinar no existe, la clase File no devuelve una excepción. Habrá que utilizar el método **exists**. Este método recibe **true** si la carpeta o archivo es válido (puede provocar excepciones *SecurityException*).

También se puede construir un objeto **File** a partir de un objeto **URL**.

### 4.11.2.- El problema de las rutas.-

Cuando se crean programas en Java hay que tener muy presente que no siempre sabremos qué sistema operativo utilizará el usuario del programa. Esto provoca que la realización de rutas sea problemática porque la forma de denominar y recorrer rutas es distinta en cada sistema operativo.

Por ejemplo en Windows se puede utilizar la barra **/** o la doble barra invertida **** como separador de carpetas, en muchos sistemas Unix sólo es posible la primera opción. En general es mejor usar las clases **Swing** (como **JFileDialog**) para especificar rutas, ya que son clases en las que la ruta de elige desde un cuadro y, sobre todo, son independientes de la plataforma.

También se pueden utilizar las **variables estáticas** que posee File. Estas son:

| propiedad | uso |
|---|---|
| char **separatorChar** | El carácter separador de nombres de archivo y carpetas. En Linux/Unix es "/" y en Windows es "\", que se debe escribir como \\, ya que el carácter \ permite colocar caracteres de control, de ahí que haya que usar la doble barra. |
| String **separator** | Como el anterior pero en forma de String |
| char **pathSeparatorChar** | El carácter separador de rutas de archivo que permite poner más de un archivo en una ruta. En Linux/Unix suele ser ":", en Windows es ";" |
| String **pathSeparator** | Como el anterior, pero en forma de String |

**Figura 36.- Variables Estáticas de la Clase File.**

Para poder garantizar que el separador usado es el del sistema en uso:

```
String ruta="documentos/manuales/2003/java.doc";

ruta=ruta.replace('/',File.separatorChar);
```

Normalmente no es necesaria esta comprobación ya que Windows acepta también el carácter / como separador.

**Métodos generales.**

| método | uso |
|---|---|
| boolean **isFile()** | Devuelve **true** si el objeto File es un archivo y **false** si es carpeta o si no existe. |
| boolean **renameTo(**File f2**)** | Cambia el nombre del archivo por el que posee el archivo pasado como argumento. Devuelve **true** si se pudo completar la operación. |
| boolean **delete()** | Borra el archivo y devuelve **true** si puedo hacerlo |
| long **length()** | Devuelve el tamaño del archivo en bytes |
| boolean **createNewFile()** Throws IOException | Crea un nuevo archivo basado en la ruta dada al objeto File. Hay que capturar la excepción *IOException* que ocurriría si hubo error crítico al crear el archivo.<br>Devuelve **true** si se hizo la creación del archivo vacío y **false** si ya había otro archivo con ese nombre. |

**Figura 37.- Métodos de la Clase File.**

| método | uso |
|---|---|
| String **toString()** | Para obtener la cadena descriptiva del objeto |
| boolean **exists()** | Devuelve **true** si existe la carpeta o archivo. |
| boolean **canRead()** | Devuelve **true** si el archivo se puede leer |
| boolean **canWrite()** | Devuelve **true** si el archivo se puede escribir |
| boolean **isHidden()** | Devuelve **true** si el objeto File es oculto |
| boolean **isAbsolute()** | Devuelve **true** si la ruta indicada en el objeto File es absoluta |
| boolean **equals(File** f2**)** | Compara f2 con el objeto **File** y devuelve verdadero si son iguales. |
| String **getAbsolutePath()** | Devuelve una cadena con la ruta absoluta al objeto File. |
| File **getAbsoluteFile()** | Como la anterior pero el resultado es un objeto File |
| String **getName()** | Devuelve el nombre del objeto File. |
| String **getParent()** | Devuelve el nombre de su carpeta superior si la hay y si no **null** |
| File **getParentFile()** | Como la anterior pero la respuesta se obtiene en forma de objeto File. |
| boolean **setReadOnly()** | Activa el atributo de sólo lectura en la carpeta o archivo. |

| | |
|---|---|
| URL **toURL()** throws MalformedURLException | Convierte el archivo a su notación URL correspondiente |
| URI **toURI()** | Convierte el archivo a su notación URI correspondiente |

**Figura 38.- Métodos de Uso General de la Clase File.**

### 4.11.3.- Secuencias de Archivo.-

**Lectura y escritura byte a byte.**

Para leer y escribir datos a archivos, Java utiliza dos clases especializadas que leen y escriben orientando a byte (Véase tema anterior); son **FileInputStream** (para la lectura) y **FileOutputStream** (para la escritura).

Se crean objetos de este tipo construyendo con un parámetro que puede ser una ruta o un objeto File:

```
FileInputStream fis=new FileInputStream(objetoFile);

FileInputStream fos=new FileInputStream(

"                                "/textos/texto25.txt");
```

La construcción de objetos **FileOutputStream** se hace igual, pero además se puede indicar un segundo parámetro booleano que con valor **true** permite añadir más datos al archivo (normalmente al escribir se borra el contenido del archivo, valor **false**).

Estos constructores intentan abrir el archivo, generando una excepción del tipo **FileNotFoundException** si el archivo no existiera u ocurriera un error en la apertura. Los métodos de lectura y escritura de estas clases son los heredados de las clases InputStream y OutputStream. Los métodos **read** y **write** son los que permiten leer y escribir. El método **read** devuelve -1 en caso de llegar al final del archivo.

Otra posibilidad, más interesante, es utilizar las clases DataInputStream y DataOutputStream. Estas clases están mucho más preparadas para escribir datos de todo tipo.

**Escritura.-**

El proceso sería:

1. Crear un objeto **FileOutputStream** a partir de un objeto **File** que posee la ruta al archivo que se desea escribir.
2. Crear un objeto **DataOutputStream** asociado al objeto anterior. Esto se realiza en la construcción de este objeto.
3. Usar el objeto del punto 2 para escribir los datos mediante los métodos **write** donde *tipo* es el tipo de datos a escribir (Int, Double). A este método se le pasa como único argumento los datos a escribir.
4. Se cierra el archivo mediante el método **close** del objeto DataOutputStream**.**

**Lectura.-**

El proceso es análogo. Sólo que hay que tener en cuenta que al leer se puede alcanzar el final del archivo. Al llegar al final del archivo, se produce una excepción del

tipo **EOFException** (que es subclase de **IOException**), por lo que habrá que controlarla.

En este listado, obsérvese como el bucle **while** que da lugar a la lectura se ejecuta indefinidamente (no se pone como condición a secas **true** porque casi ningún compilador lo acepta), se saldrá de ese bucle cuando ocurra la excepción EOFException que indicará el fin de archivo.

Las clases DataStream son muy adecuadas para colocar datos binarios en los archivos.

**Lectura y escritura mediante caracteres.-**

Como ocurría con la entrada estándar, se puede convertir un objeto **FileInputStream** o **FileOutputStream** a forma de **Reader** o **Writer** mediante las clases **InputStreamReader** y **OutputStreamWriter**.

Existen además dos clases que manejan caracteres en lugar de bytes (lo que hace más cómodo su manejo), son **FileWriter** y **FileReader.** La construcción de objetos del tipo FileReader se hace con un parámetro que puede ser un objeto File o un String que representarán a un determinado archivo.

La construcción de objetos FileWriter se hace igual sólo que se puede añadir un segundo parámetro booleano que, en caso de valer **true**, indica que se abre el archivo para añadir datos; en caso contrario se abriría para grabar desde cero (se borraría su contenido).

Para escribir se utiliza **write** que es un método void que recibe como parámetro lo que se desea escribir en formato int, String o array de caracteres. Para leer se utiliza el método read que devuelve un int y que puede recibir un array de caracteres en el que se almacenaría lo que se desea leer. Ambos métodos pueden provocar excepciones de tipo IOException.

## 4.12 Serialización.-

Es una forma automática de guardar y cargar el estado de un objeto. Se basa en la interfaz **serializable** que es la que permite esta operación. Si un objeto ejecuta esta interfaz puede ser guardado y restaurado mediante una secuencia.

La *serialización de objetos* de Java nos permite tomar cualquier objeto que implemente la interfase **Serializable** y convertirlo en una secuencia de bytes que pueda más tarde ser totalmente restaurada para regenerar el objeto original. Esto es verdad inclusive a través de la red, lo que significa que el mecanismo de serialización compensa automáticamente las diferencias de los sistemas operativos. Esto es, se puede crear un objeto en una máquina Windows, serializarlo, y enviarlo a través de la red a una máquina Unix donde será correctamente reconstruido.

No nos tenemos que preocupar por la representación de los datos en las diferentes máquinas, por el orden de los bytes o cualquier otro detalle. Por si mismo, la serialización de objetos es interesante porque permite implementar *persistencia ligera*.

Recuerde que esta persistencia significa que el tiempo de vida de un objeto no está determinado por si un programa es ejecutado o no el objeto vive *entre invocaciones* del programa. Tomando un objeto serializable y escribiéndolo en el disco, luego restaurándolo cuando el programa es invocado nuevamente, se es capaz de producir el efecto de persistencia. La razón por la cual es llamado "persistencia ligera" es que no se puede simplemente definir un objeto utilizando algún tipo de palabra clave "persistente" y dejar que el sistema se haga cargo de los detalles (a pesar de que esto puede suceder en el futuro). En lugar de eso, se debe explícitamente serializar y deserializar los objetos en su programa.

Cuando se desea utilizar un objeto para ser almacenado con esta técnica, debe ser incluida la instrucción **implements** Serializable (además de importar la clase **java.io.Serializable**) en la cabecera de clase. Esta interfaz no posee métodos, pero es un requisito obligatorio para hacer que el objeto sea serializable.

La clase **ObjectInputStream** y la clase **ObjectOutputStream** se encargan de realizar estos procesos. Son las encargadas de escribir o leer el objeto de un archivo. Son herederas de **InputStream** y **OutputStream**, de hecho son casi iguales a **DataInput/OutputStream** sólo que incorporan los métodos **readObject** y **writeObject** que son muy poderosos.

El listado anterior podría ser el código de lectura de un archivo que guarda coches. Los métodos **readObject** y **writeObject** usan objetos de tipo Object, readObject les devuelve y writeObject les recibe como parámetro. Ambos métodos lanzan excepciones del tipo **IOException** y **readObject** además lanza excepciones del tipo **ClassNotFoundException**.

## 4.13 Generación de Código en la Aplicación Swing Evolution.-

# Introducción.-

Swing Evolution como uno de sus objetivos primordiales está la generación de los componentes creados por el usuario por lo tanto es una de las partes fundamentales que genere un código que se apegue estrictamente a las especificaciones de la sintaxis de Java y a las especificaciones de un bean puesto que el componente tiene que acoplarse al IDE NetBeans.

Para la generación de código Swing Evolution trabaja directamente con archivos ya sea el que está generando (.java) o los archivos de la carpeta lib(librerías de Swing Evolution con extensión **.se** que fueron creadas por los autores de este proyecto) y también genera código desde las clases de Swing Evolution según los valores de los atributos de las clases que corresponden al componente que se está creando(ButtonUI, TextFieldUI, PasswordFieldUI, etc.).

# Las Librerías.-

Swing Evolution tiene librerías con la extensión **.se** que no son más que archivos de texto formateado que contiene patrones de código que se los utiliza para la generación de código (archivos .java) de los nuevos componentes swing. En las librerías tenemos ciertas líneas que le permiten a la aplicación ubicarse dentro del fichero y saber dónde va a insertar el código de la librería al archivo .java que se está generando como por ejemplo:

Por ejemplo esta librería llamada reflejar.se sirve para insertar el método reflejar en la clase de un JPanel que tiene el efecto de reflexión.

**Código de la librería reflejar.se**

```
//importaciones
```

```
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import javax.swing.JComponent;
import javax.swing.JPanel;
//fin_importaciones

//atributos
private float altura;
private float opacidad;
private ReflectionPanel reflectionPanel;
private JComponent componente;
//fin_atributos

//clase
        public void reflejar(JComponent componente){
        this.setComponente(componente);
        setLayout(new FlowLayout(FlowLayout.CENTER,1,1));
        add(buildReflectionPanel());
    }


    private JComponent buildReflectedComponent() {
        try {
            Class.forName("quicktime.QTSession");
        } catch (ClassNotFoundException ex) {
            return getComponente();
        }
```

```
        return null;
    }

    private JComponent buildReflectionPanel() {
        setReflectionPanel(new ReflectionPanel(altura,opacidad));
        getReflectionPanel().add(buildReflectedComponent());
        return getReflectionPanel();
    }

//fin_clase
```

Como se puede observar hay líneas documentadas que le sirven a Swing Evolution saber de dónde hasta dónde están estos bloques de código como son las importaciones, atributos, y el método que van dentro de las líneas documentadas clase y fin_clase para poder transcribir desde la librería hacia la clase .java que se está generando

Como en el ejemplo se observa la primera y la última línea hacen de limitador del código que está entre ellas, es como así el sistema sabe donde ubicar el código de la librería en la o clases que está generando.

**Donde Empieza La Generación de Código.-**

Todo el proceso que ejecuta Swing Evolution para generar código es el siguiente:

- Todo empieza en las clases de Swing Evolution que representan los componentes que se va a modificar (ButtonUI, LabelUI, etc., del paquete se.proceso.componentes), en donde habiendo el usuario modificado al componente tienen estas clase de cómo se debe generar el código fuente.
- La clase que representa al componente utiliza una clase llamada RecolectorDatos que ayuda a ingresar sentencias en un vector.

- Luego que está recolectado en este Vector todas las instrucciones necesarias para crear el código, este Vector se envía a una clase intermedia entre los componentes y las clases de generación de código llamada MedCodigo, la misma que lee una a una las instrucciones que están en el Vector de instrucciones y las va ejecutando pasándole el control a una Clase llamada Código.

  La clase Código es la encargada de llamar métodos de la clase ArchivoJava la misma que se encarga de crear, accesar, leer y escribir en los archivos .java y también las librerías .se.

A continuación se presenta el resultado de la generación de código de Swing Evolution:

**//Código Generado por Swing Evolution**

```
package javax.swing.se.BotonNormal;


//importaciones
import javax.swing.JButton;
import java.awt.Dimension;
import javax.swing.Icon;
import java.awt.Rectangle;
import java.awt.Shape;
```

```
import java.awt.Color;
import java.io.Serializable;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.GradientPaint;
import java.awt.geom.RoundRectangle2D;
//fin_importaciones

//clase
public class BtNormal extends JButton {

	//atributos
	private Color color_normal1=new Color(0,0,0);;
	private Color color_normal2=new Color(255,0,0);;
	private Color color_click1=new Color(0,0,255);;
	private Color color_click2=new Color(0,255,255);;
	private transient GradientPaint gradiante;
	private int achatamiento_x=30;
	private int achatamiento_y=30;
	//fin_atributos

	//constructor
	public BtNormal(){
		inicializarTodo();
	}
	//fin_constructor



//inicializarTodo
```

```
      public void inicializarTodo(){
            setContentAreaFilled(false);
            setBorderPainted(true);
            setFocusPainted(true);
      }
//fin_inicializarTodo


//paint
public void paintComponent(Graphics g){


Graphics2D lapiz=(Graphics2D)g;
lapiz.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.
VALUE_ANTIALIAS_ON);
RoundRectangle2D.Float r2d = new RoundRectangle2D
                                                .Float(0,0,getWidth(),
                                                getHeight(),
                                                getAchatamiento_x(),
                                                getAchatamiento_y());
lapiz.clip(r2d);
if(getModel().isArmed())
      gradiante=new GradientPaint(0,0,
                                  color_click1,
                                  0,getHeight(),
                                  color_click2);
else
      gradiante=new GradientPaint(0,0,
                                  color_normal1,0,
                                  getHeight(),
                                  color_normal2);
      lapiz.setPaint(gradiante);
```

```
        lapiz.drawRoundRect(0, 0,
        getWidth(),getHeight(),getAchatamiento_x(),getAchatamiento_y());

        lapiz.fillRoundRect(0, 0,
        getWidth(),getHeight(),getAchatamiento_x(),getAchatamiento_y());

        lapiz.setColor(getForeground());

        super.paintComponent(lapiz);

    }

    //fin_paint


        public void setColor_normal1(Color color_normal1){

              this.color_normal1=color_normal1;

        }


        public Color getColor_normal1(){

              return this.color_normal1;

        }


        public void setColor_normal2(Color color_normal2){

              this.color_normal2=color_normal2;

        }


        public Color getColor_normal2(){

              return this.color_normal2;

        }


        public void setColor_click1(Color color_click1){

              this.color_click1=color_click1;

        }


        public Color getColor_click1(){
```

```
        return this.color_click1;
    }

    public void setColor_click2(Color color_click2){
        this.color_click2=color_click2;
    }

    public Color getColor_click2(){
        return this.color_click2;
    }

    public void setGradiante(GradientPaint gradiante){
        this.gradiante=gradiante;
    }

    public GradientPaint getGradiante(){
        return this.gradiante;
    }

    public void setAchatamiento_x(int achatamiento_x){
        this.achatamiento_x=achatamiento_x;
    }

    public int getAchatamiento_x(){
        return this.achatamiento_x;
    }

    public void setAchatamiento_y(int achatamiento_y){
        this.achatamiento_y=achatamiento_y;
    }
```

```
	public int getAchatamiento_y(){
		return this.achatamiento_y;
	}


}
//fin_clase
```

**//Fin Código Generado por Swing Evolution**

**Breve explicación de las instrucciones que puede ejecutar Swing Evolution.**

Entre las sentencias que Swing Evolution puede realizar son:

**modificar_cadena.-** Sirve para modificar las cadenas de texto dentro de la clase que se está generando como por ejemplo los valores de una variable.

**insertar_codigo.-** Inserta código en la clase que se está generando en un determinado lugar.

**transcribir_codigo.-** Copia código de una librería .se hacia la clase que se está generando

**crear_instancia.-** Crea línea de código para efectuar una instancia en un determinado lugar de la clase que se está generando.

**llamar_metodo.-** Crea línea de código para efectuar la llamada a un método y la ubica en un determinado lugar de la clase que se está generando.

**transcribir_clase.-** Copia código intacto desde un archivo .se a la clase que se está generando pero no genera los métodos set y get de los atributos, los cuales son primordiales para un bean.

**guardar_figura.-** Crea otra clase a la principal generando métodos set y get de los atributos, y transcribe código pertinente en la misma.

**agregar_interface.-** Agrega la interface en la declaración de la clase que se está generando, y crea la interface .java desde la librería .se.

# 5. EVALUACIÓN DEL OBJETO DE INVESTIGACIÓN

La Carrera de Ingeniería en Sistemas del Área de Energía y Recursos Naturales no Renovables de la Universidad Nacional de Loja, forma profesionales con conocimientos profundos en la estructura y particularidades del software, capaces de llevar a la práctica todos los conocimientos adquiridos durante los seis años de carrera universitaria, mediante la realización de una Herramienta Visual que permita la creación de nuevos componentes swing utilizando Java2D y 3D.

Debido a la no existencia de herramientas visuales (IDE) para mejorar la apariencia de los controles Swing Java, se identificó que el objeto de investigación es la " No existencia de un Entorno de Desarrollo Integrado (IDE) para la personalización de los controles Swing de JAVA", por ello se decidió diseñar e implementar una herramienta visual para la creación de nuevos componentes Swing utilizando Java2D y 3D, usando como base principal del proyecto la plataforma Java y los APIS Java2D, lo que permitirá tanto a los programadores mejorar de gran manera la apariencia visual de estos controles y a su vez dar una aplicación atractiva para el usuario final, solucionando efectivamente esta problemática.

Cabe destacar que para cumplir con los objetivos planteados se elaboró un análisis completo de la problemática, viendo las ventajas y desventajas que este podría presentar en el desarrollo del presente proyecto de Tesis, llegando así a la culminación satisfactoria de esta herramienta de programación.

# 6. DESARROLLO DE LA PROPUESTA ALTERNATIVA

## 6.1. ANÁLISIS DE LA APLICACIÓN

### 6.1.1. Definición del problema

En la actualidad, los problemas más comunes que los programadores tienen a la hora de desarrollar software en la plataforma Java están vinculados directamente a la Vista, GUI (Interfaz Gráfica de Usuario). Es así que en el mundo del software los componentes Swing de la plataforma Java ayudan de alguna manera a los programadores a mejorar la creación de Interfaz Gráfica de Usuario para así poder tener un producto amigable acorde a las exigencias del usuario.

Los problemas más relevantes que radican en este tipo de componentes JFC/SWING es la pérdida de tiempo a la hora de su personalización ya que los desarrolladores de la plataforma Java deben acoplarse a la apariencia normal básica del J2SE (JDK Máquina Virtual de Java), es por esta razón que el tiempo de desarrollo para la personalización de cada uno de los componentes Swing se hace cada vez más difícil. Así mismo los programadores tienen que escribir muchas líneas de código para poder lograr obtener una personalización eficaz de los componentes.

Un problema que también se presentan muy común en el proceso de la personalización de los componentes SWING es la escritura del código fuente, el mismo que en algunos casos no puede ser entendido de manera rápida por los programadores lo cual les ocasiona confusión en el proceso de aprendizaje.

En la actualidad las herramientas que están manejando los programadores de la plataforma Java se han encargado de la personalización de la GUI (Interfaz Gráfica de Usuario) de manera superficial y básica ,es por ello que vemos la necesidad de ***"Diseñar e implementar una herramienta Visual que permita la creación de nuevos componentes Swing utilizando las tecnología Java2D y 3D".***

### 6.1.2. Glosario de Términos

| Nombre | Descripción | Visibilidad |
|---|---|---|
| Plataforma Swing Evolution | Herramienta de programación, utilizada para mejorar el tiempo de desarrollo en el proceso de creación de software. | Visible |
| Control Swing | Componente de Software (Java Bean) utilizado para la interacción con el usuario final. | Visible |
| Paleta de Proyectos | Componente gráfico de la aplicación utilizado para la organización de los proyectos creados dentro de la plataforma (Swing Evolution). | Visible |
| Paleta Plantillas | Componente gráfico de la aplicación utilizado para la organización de plantillas. | Visible |
| Paleta Combinación | Componente gráfico utilizado para realizar la manipulación de las formas (figuras básicas) atreves de una operación lógica. | Visible |
| Paleta de Información | Componente gráfico utilizado para la generación de reportes de las tareas que la aplicación ejecuta en tiempo de ejecución. | Visible |

| | | |
|---|---|---|
| Paleta de Propiedades | Componente gráfico que se utiliza para la extracción de las propiedades encapsulas dentro de los componentes Swing, a través de las técnicas de los JavaBeans. | Visible |
| JavaBean | Es un componente de Software reutilizable. | Oculto |
| Componente | Es un control swing que tiene las propiedades de un componente JavaBean. | Visible |
| Paleta Efectos | Componente Gráfico que se utiliza para aplicar efectos (Apariencia 2D & 3D) a cada unos de los componentes Swing. | Visible |
| Programador | Es la persona que tiene conocimientos de los lenguajes de programación. | Oculto |
| Diseñador | Componente gráfico en el cual se visualiza el componente que se está creando en tiempo de diseño. | Visible |
| Paleta Formas Básicas | Componente gráfico que se utiliza para organizar las diferentes formas (Cirulo, Elipse, Estrella, Hexágono, Nonágono….etc.) aplicables para los componentes. | Visible |
| Paleta Componentes Swing | Componente Gráfico que se encarga de organizar cada uno de los controles Swing JAVA. | Visible |
| Paleta de Información. | Componente gráfico que se utiliza para dar a conocer todo lo que sucede dentro de la creación del proyecto. | Visible |
| Wizard | Componente gráfico que visualiza un cuadro de opciones de los componentes Swing para su personalización. | Visible |

### 6.1.3. Documento de Requerimientos

#### 6.1.3.1. Requerimientos Funcionales

| N° Req | Descripción | Tipo |
|---|---|---|
| RF01 | El sistema cambiará la forma de los componentes Swing utilizando las tecnologías Java 2D y 3D | Visible |
| RF02 | El sistema agregará efectos 2D y 3D en los componentes Swing. | Visible |
| RF03 | El sistema generará código del o de los componentes creados en el proyecto. | Oculto |
| RF04 | El sistema permitirá la instalación de los nuevos componentes Swing a la plataforma NetBeans versión 5.0 y posteriores. | Oculto |
| RF05 | El sistema permitirá al usuario la creación de Nuevos Proyectos. | Visible |
| RF06 | El sistema permitirá al usuario Guardar los Proyectos realizados. | Visible |
| RF07 | El sistema permitirá al usuario abrir proyectos de los componentes anteriormente creados. | Visible |
| RF08 | El sistema permitirá al usuario la eliminación de proyectos. | Visible |
| RF09 | El sistema permitirá al usuario crear, guardar y abrir plantillas de los nuevos componentes. | Visible |
| RF10 | El sistema permitirá deshacer y rehacer (Ctrl+Z y Ctrl+Y) los cambios realizados durante la creación del nuevo componente. | Oculto |
| RF11 | El sistema permitirá crear nuevos componentes a través de las combinaciones de las formas básicas (Círculos, Elipses, Cuadrados, Rectángulos y plantillas), combinaciones y operaciones de conjuntos. | Visible |
| RF12 | El sistema permitirá realizar la compilación en caliente de cada unos de los componentes creados. | Oculto |

| | | |
|---|---|---|
| RF13 | El sistema permitirá generará paquetes de los nuevos componentes creados. | Oculto |
| RF14 | El sistema permitirá crear una estructura de directorios y archivos para el proyecto que será creado | Oculto |
| RF15 | El sistema permitirá crear una estructura de directorios y archivos para las plantillas que serán creadas. | Oculto |
| RF16 | El sistema permitirá la creación de un ejecutable .JAR para cada uno de los componentes creados dentro de un proyecto. | Visible |

#### 6.1.3.2. Requerimientos no Funcionales

| Código | Atributo | Descripción |
|---|---|---|
| **R2.1** | Interfaz gráfica | • La distribución de los componentes visuales de la aplicación están representados por Paletas.<br>• La aplicación mostrará las diferentes paletas de proyectos y plantillas para su manejo visual a travéz del mouse. |
| **R2.2** | Tiempo de respuesta | • La aplicación está diseñada para trabajar de la manera más eficiente con la ejecución de un proyecto a la vez, mejorando así su rendimiento. |
| **R2.3** | Tolerancia a fallos | • En la aplicación está controlada todo tipo de excepciones tanto en la parte de diseño como en la de ejecución. |
| **R2.4** | Rendimiento de la aplicación. | • El rendimiento está vinculado al número de componentes personalizados por cada proyecto creado dentro de la herramienta. |
| **R2.6** | Facilidad de uso | • La herramienta presenta una distribución de sus componentes gráficos amigable a los usuarios para su manejo. |

## 6.2. MODELADO DEL DOMINIO

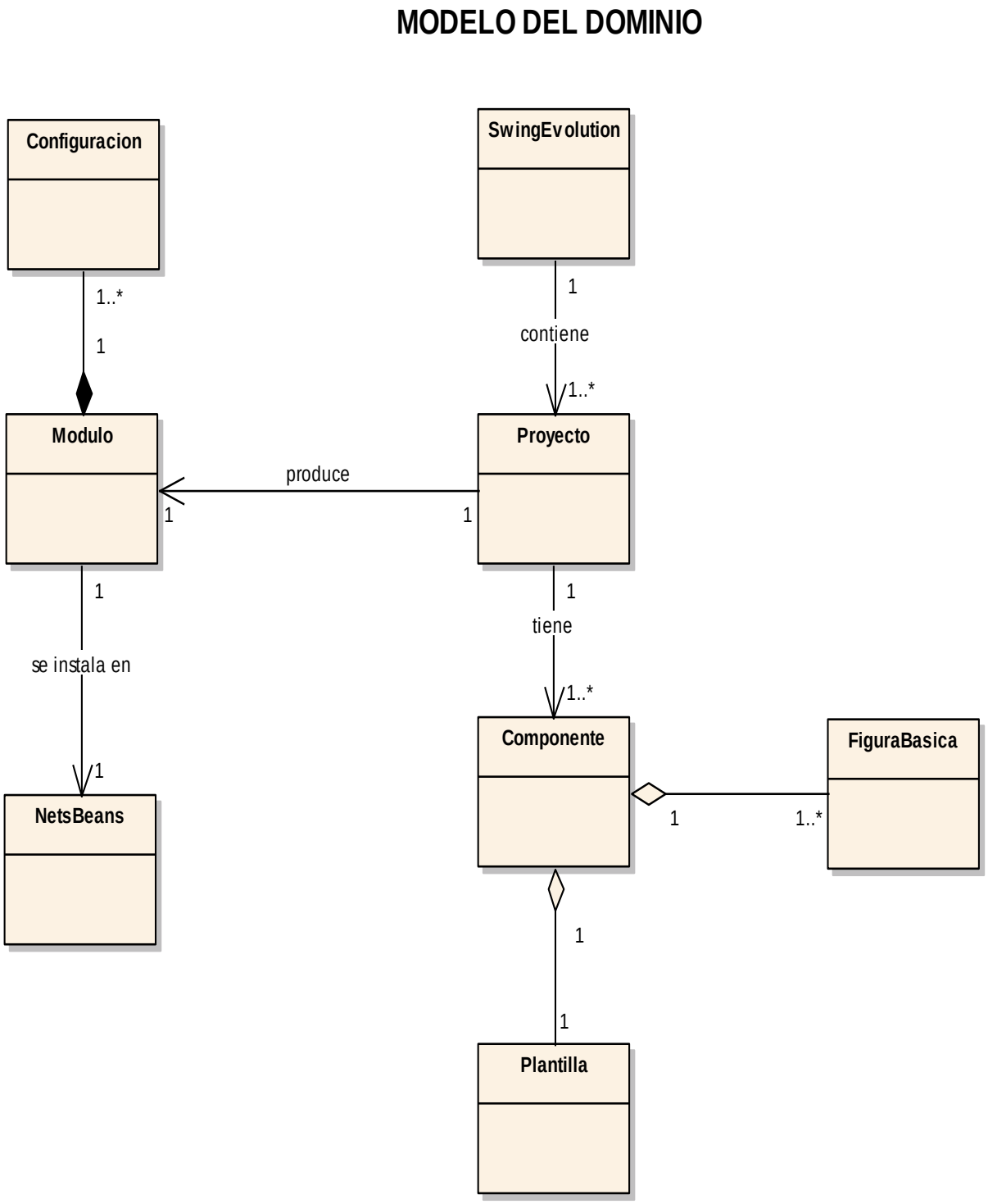



**Figura 1. - Modelo Del Dominio.**

## 6.3. Modelo Dinámico de la aplicación

### 6.3.1. Diagrama de Casos de Uso

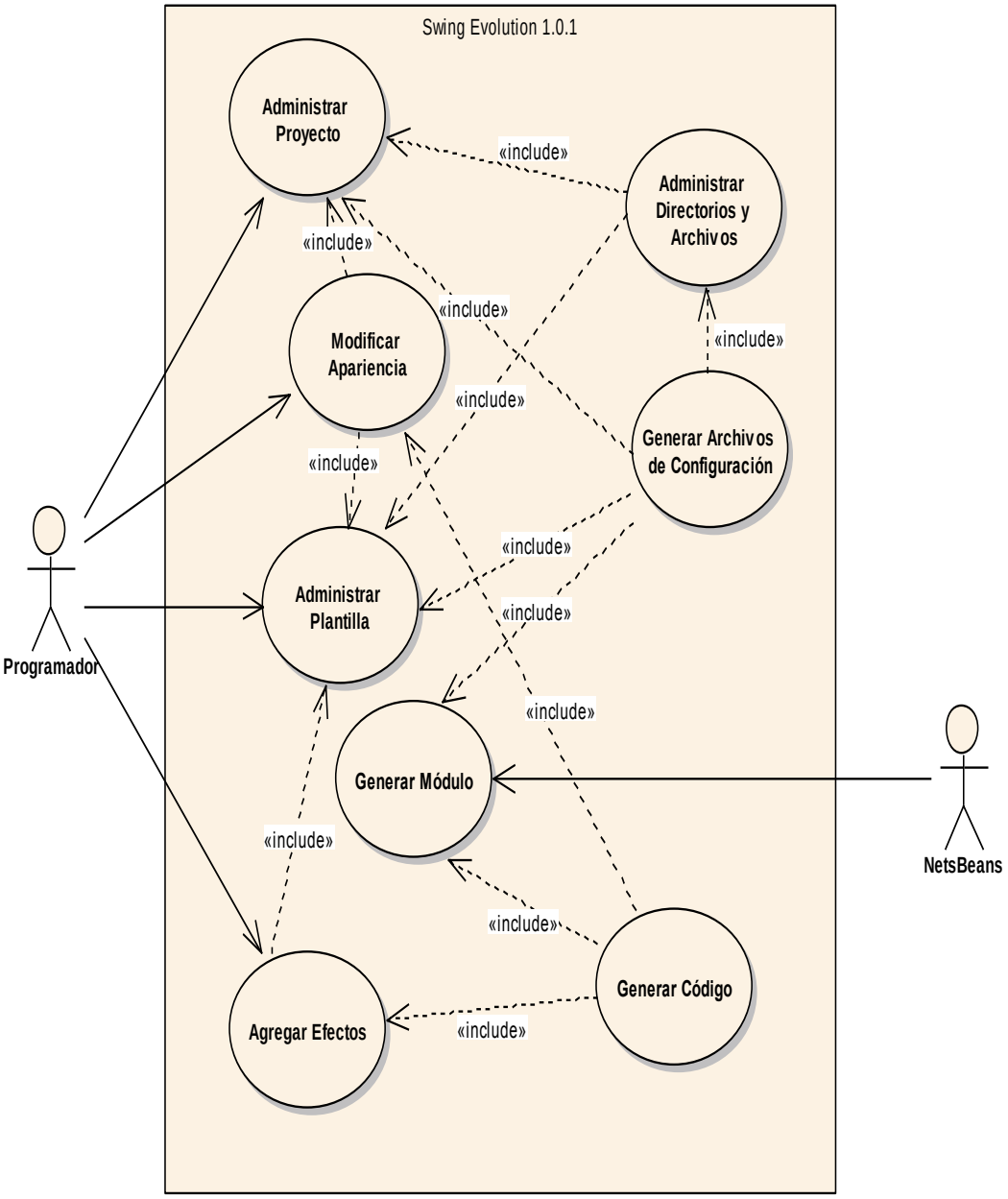



Figura 2.- Diagrama de Casos de Uso

### 6.3.2. Descripción de casos de uso

#### 6.3.2.1. Use Case: Administrar Proyecto.

| | |
|---|---|
| **Nombre:** | **Administrar Proyecto** |
| **Actor(es):** | Programador, Netbeans. |
| **Propósito:** | Permitir Crear, abrir, guardar, eliminar, deshacer y rehacer un proyecto. |
| **Visión General** | Proporcionar al programador una forma fácil y sencilla de acceder a los proyectos, para su respectiva modificación, creación o eliminación de un nuevo proyecto. |
| **Tipo:** | Primario, esencial |
| **Referencias:** | RF05, RF06, RF07, RF08, RF10. |
| **Precondición(es)** | |
| **Post-condiciones(es)** | Que se genere el ejecutable .jar |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Programador** | **Sistema** |
| 1.- El use case inicia cuando el programador elige en el menú principal de la aplicación la opción "Nuevo Proyecto" | 2.- Presenta en el Wizard "Nuevo Proyecto" |
| 3.- El programador elige el componente a personalizar. | 4.-En el dialogo Wizard se activa el botón siguiente, presentando las opciones sobre el proyecto como es: el nombre del proyecto, nombre del paquete, nombre del componente JavaBean, ubicación del proyecto etc. |
| 5.- Ingresa los datos solicitados por el sistema | 6.- Guarda los datos ingresados por el programador.<br>7.- Genera la estructura del proyecto.<br>8.- Ejecuta el use case "Administrar Directorios y Archivos". |

| | |
|---|---|
| | 9.- Ejecuta el Use Case “Generar Código”.<br><br>10.- Informar el estado sobre las acciones del Use Case “Administrar Directorios y Archivos”<br><br>11.- El sistema presenta el Diseñador con el componente a personalizar.<br><br>12.- Presentar la Paleta de Propiedades y Paleta de Formas Básicas del componente JAVA Bean a personalizar. |
| 13.- Elige la opción “Clean & Build”, en el dialogo de Opciones del Proyecto (Popup Menu) | 14.- Visualiza la paleta de Información, informando sobre las acciones realizadas para la generación del modulo .jar”.<br><br>15. Finaliza el Use Case. |
| **CURSO ALTERNO DE EVENTOS** | |
| **A.- ABRIR PROYECTO** | |
| A1.- En el paso1 del Curso Normal de eventos el programador elige la opción de Abrir Proyecto.<br><br>A3.- Selecciona el proyecto y lo abre. | A2.- El sistema presenta un cuadro de dialogo donde se debe elegir la ubicación del proyecto a abrir<br><br>A4.- Abre el proyecto seleccionado por el programador.<br><br>A5.- El sistema presenta todo el proyecto en la paleta de proyectos, para continuar con la creación de nuevos componentes.<br><br>A6.- Reconstruye el árbol de Directorios del proyecto que se desea abrir.<br><br>A7.- Regresa al paso 17 del Curso Normal de Eventos. |

| | |
|---|---|
| **B.- ELIMINAR PROYECTO** | |
| B1.- En el paso 1 del Curso Normal de Eventos el programador elige en la paleta de proyectos la opción de Eliminar un Proyecto.<br><br>B3.- El programador elige el proyecto a eliminar. | B2.- El sistema presenta un cuadro de dialogo (Popup Menu) con la opción "Delete", en la paleta proyectos.<br><br>B4.-Elimina el proyecto.<br><br>B5.- Informar sobre el proyecto que se eliminó.<br><br>B6.- Regresa al paso 17 del Curso Normal de Eventos. |
| **C.- CERRAR PROYECTO** | |
| C1. En el paso 1 del curso normal de eventos el programador elige la opción "Cerrar Proyecto" (Close) en el Popup Menu de la Paleta Proyectos.<br><br>C3.- El programador elige el proyecto a cerrar (Close). | C2.- El sistema presenta un cuadro de dialogo (Popup Menu) con la opción "Close" (Cerrar Proyecto), en la paleta proyectos.<br><br>C4. Cierra el proyecto seleccionado.<br><br>C5.- Informar sobre el proyecto que se cerró.<br><br>C6.- Regresa al paso 17 del Curso Normal de Eventos. |
| **D.- NUEVO COMPONENTE** | |
| D1.- El paso 13 del Curso Normal de Eventos el programador elige la opción "Nuevo Componente" en el Popup Menu de la Paleta de Proyectos ().<br><br>D3.- Arrastra el componente a personalizar de la Paleta de Componentes. | D2.- Visualiza el Diseñador en blanco, presenta la Paleta Componentes y Formas Básicas.<br><br>D4.- El sistema presenta un cuadro de Dialogo |

| | |
|---|---|
| | para ingresar el “Nombre del Nuevo Componente a Personalizar”.<br><br>D5.- Visualiza el Componente en el Diseñador.<br><br>D6.- Regresa al paso 13 del Curso Normal de Eventos. |
| **E.- CONSTRUIR UN PROYECTO”BUILD”** | |
| E1.- El paso 13 del Curso Normal de Eventos el programador elige la opción “Build” en el Popup Menu de la Paleta de Proyectos. | E2.- Visualiza la paleta de Información, informando sobre la realización de la Tarea “Build” seleccionada por el programador.<br><br>E3.- Regresa al paso 15 del Curso Normal de Eventos. |
| **F.- LIMPIAR PROYECTO “CLEAN”** | |
| F1.- El paso 13 del Curso Normal de Eventos el programador elige la opción “Clean” en el Popup Menu de la Paleta de Proyectos. | F2.- Visualiza la paleta de Información, informando sobre la acción Tarea “Clean” seleccionada por el programador.<br><br>F3.- Regresa al paso 15 del Curso Normal de Eventos. |

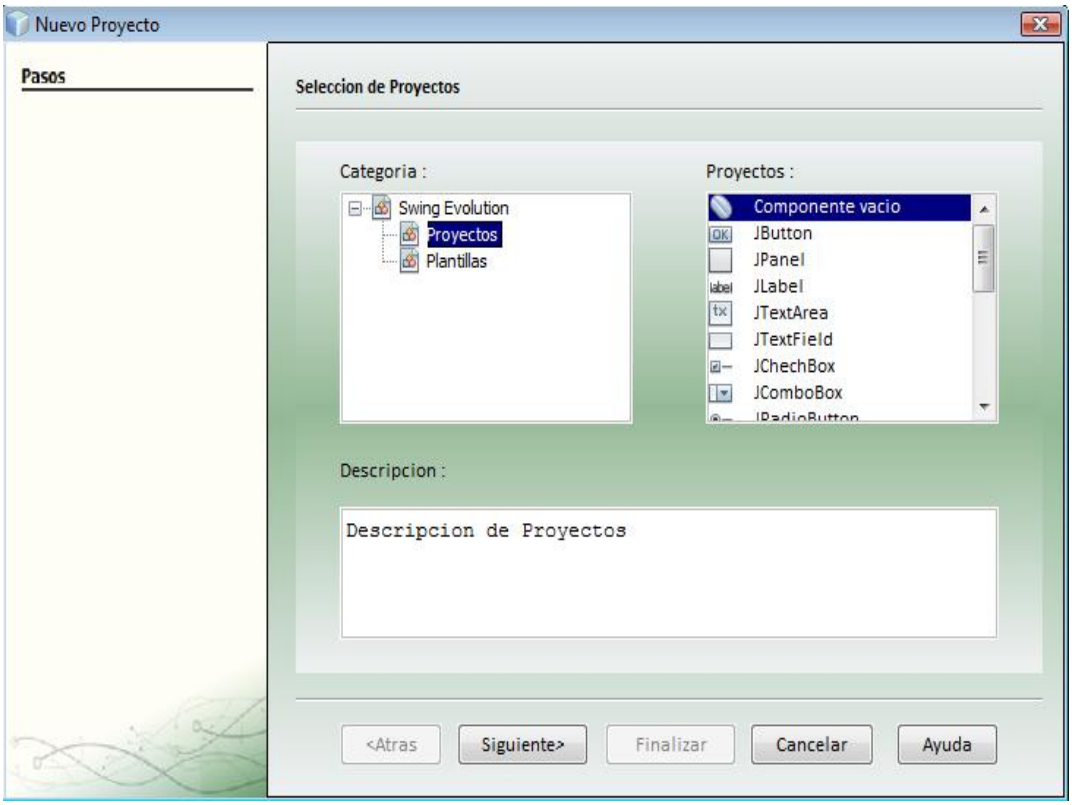



**Figura 2. Pantalla Administrar Proyecto (Nuevo Proyecto).**

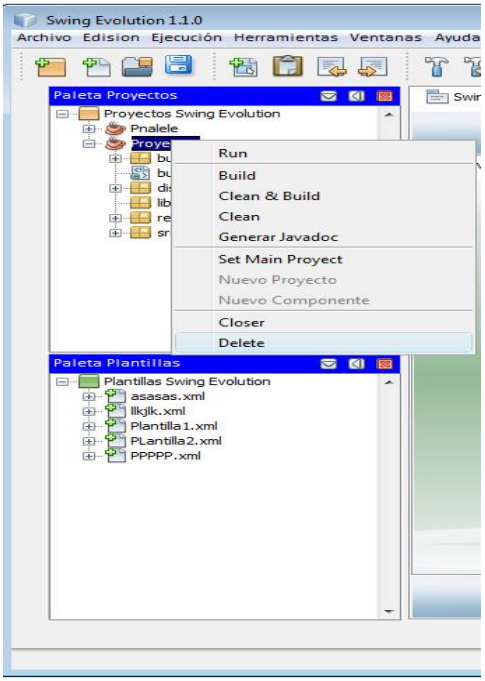



**Figura 3. Pantalla Administrar Proyecto (Eliminar Proyecto).**

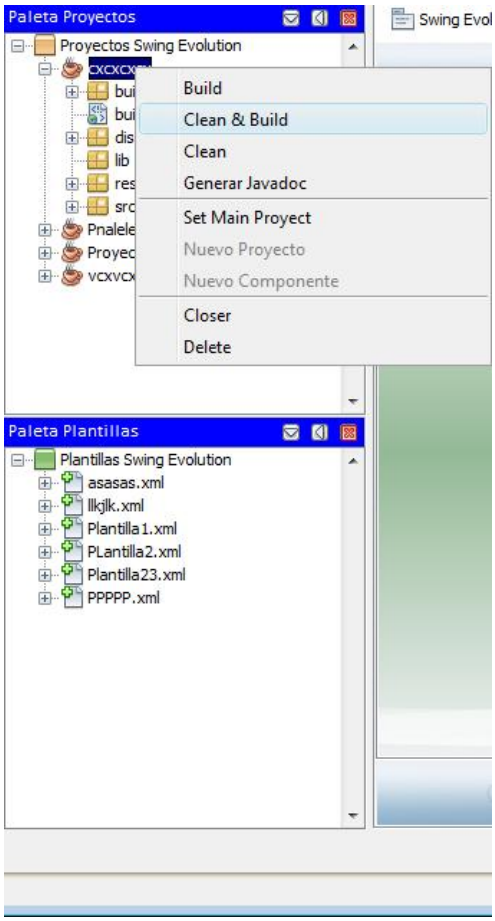



**Figura 4. Pantalla Administrar Proyecto ( Clean & Build Proyecto).**

**ADMINISTAR PROYECTO (CURSOS NORMALES)**

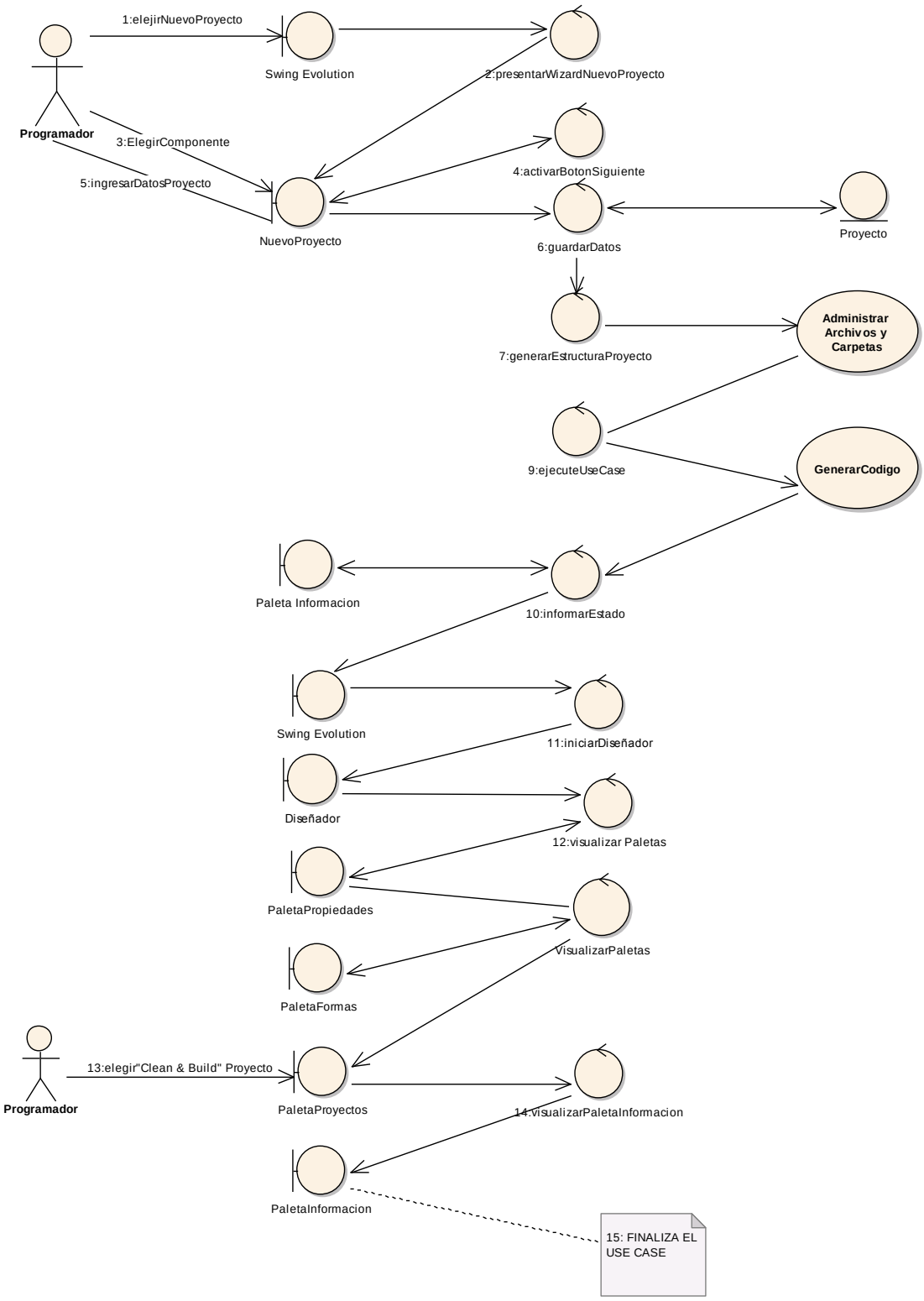



Figura 6. Use Case Administrar Proyecto

## ADMINISTRAR PROYECTO (CURSOS ALTERNOS)

## CASO ALTERNO A: ABRIR UN PROYECTO

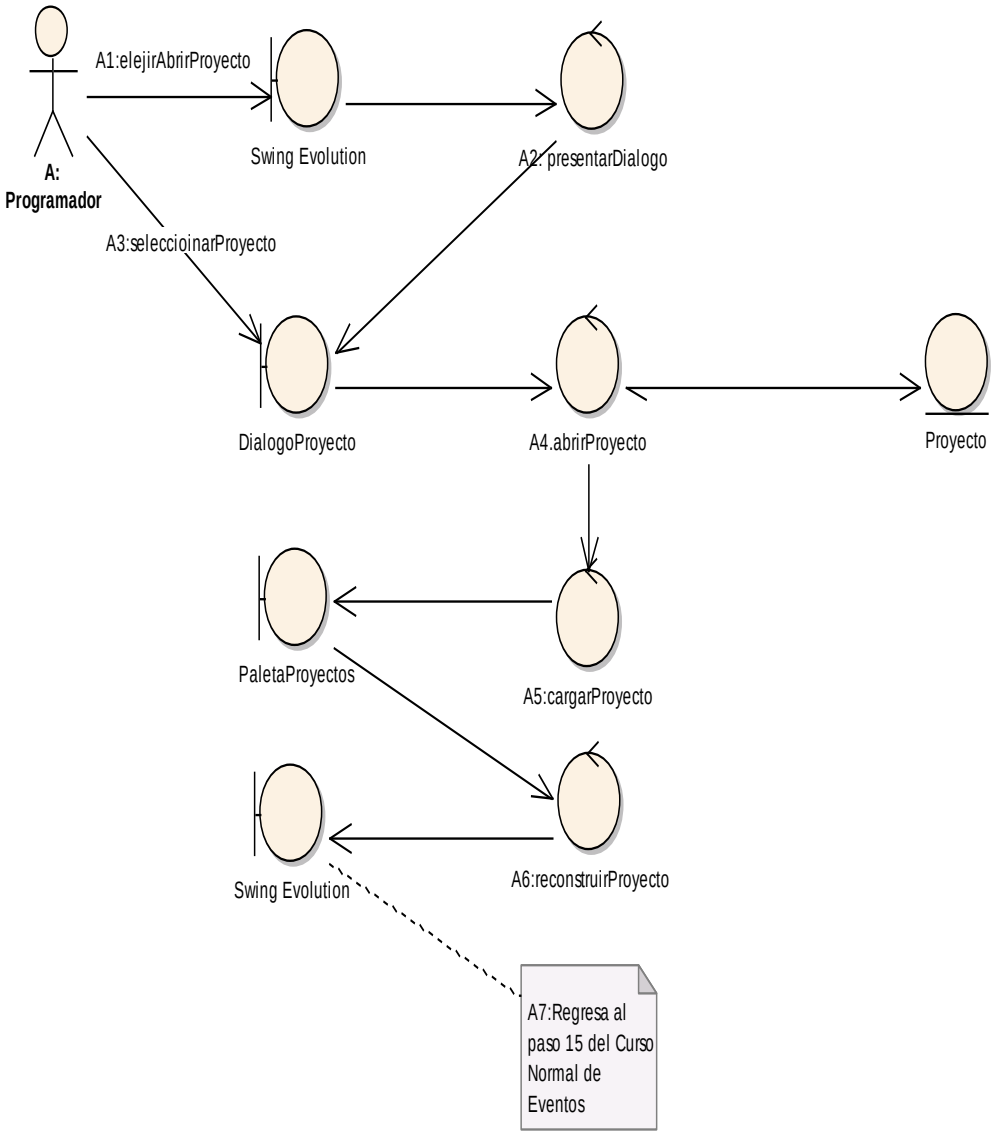



Figura 7.- Curso Alterno A: Abrir Proyecto

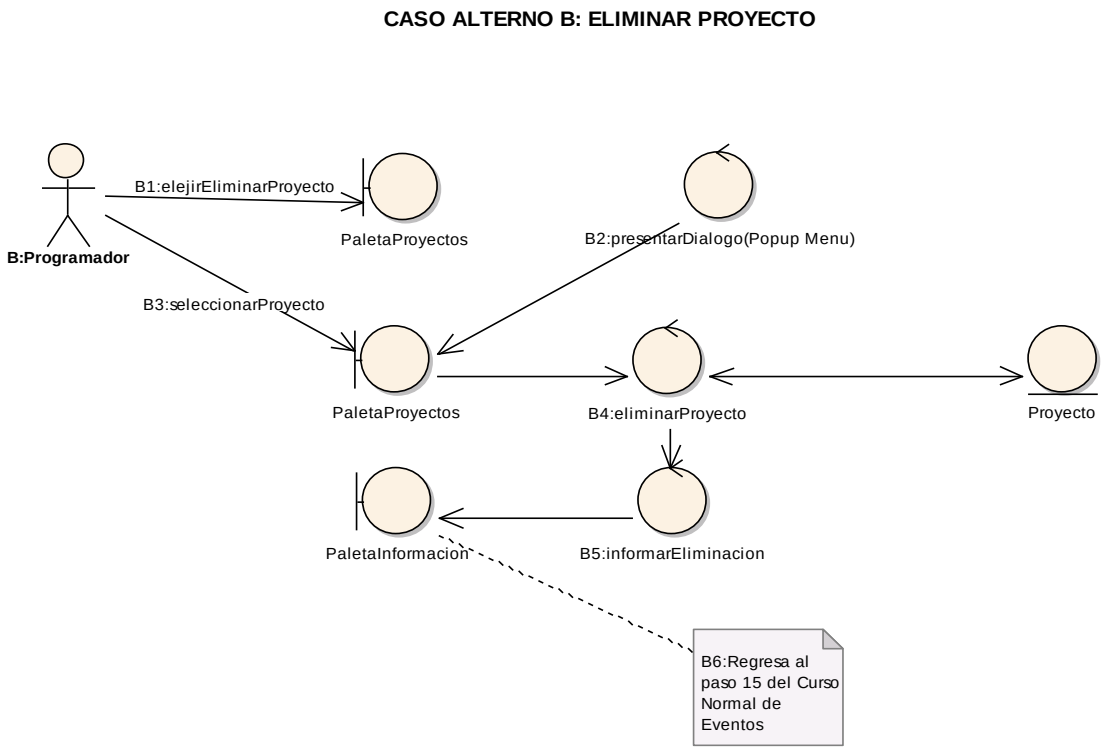



Figura 8.- Curso Alterno B: Eliminar Proyecto.

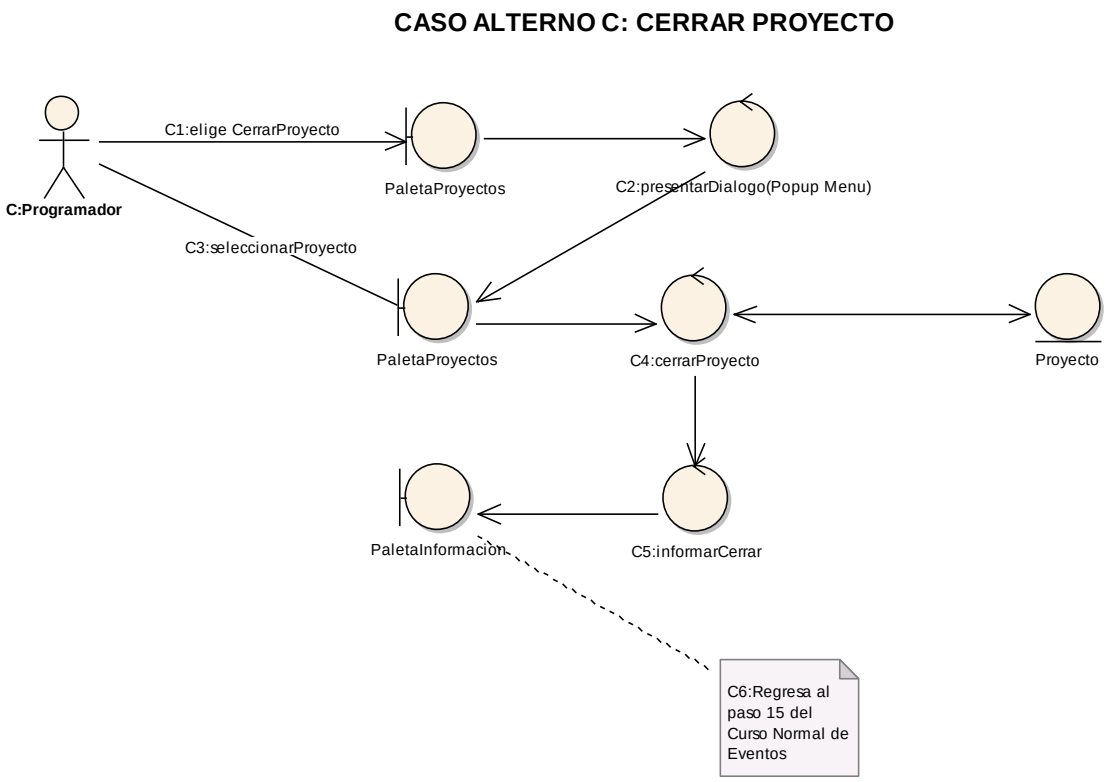


Figura 9.- Curso Alterno C: Cerrar Proyecto.

## CASO ALTERNO D: NUEVO COMPONENTE

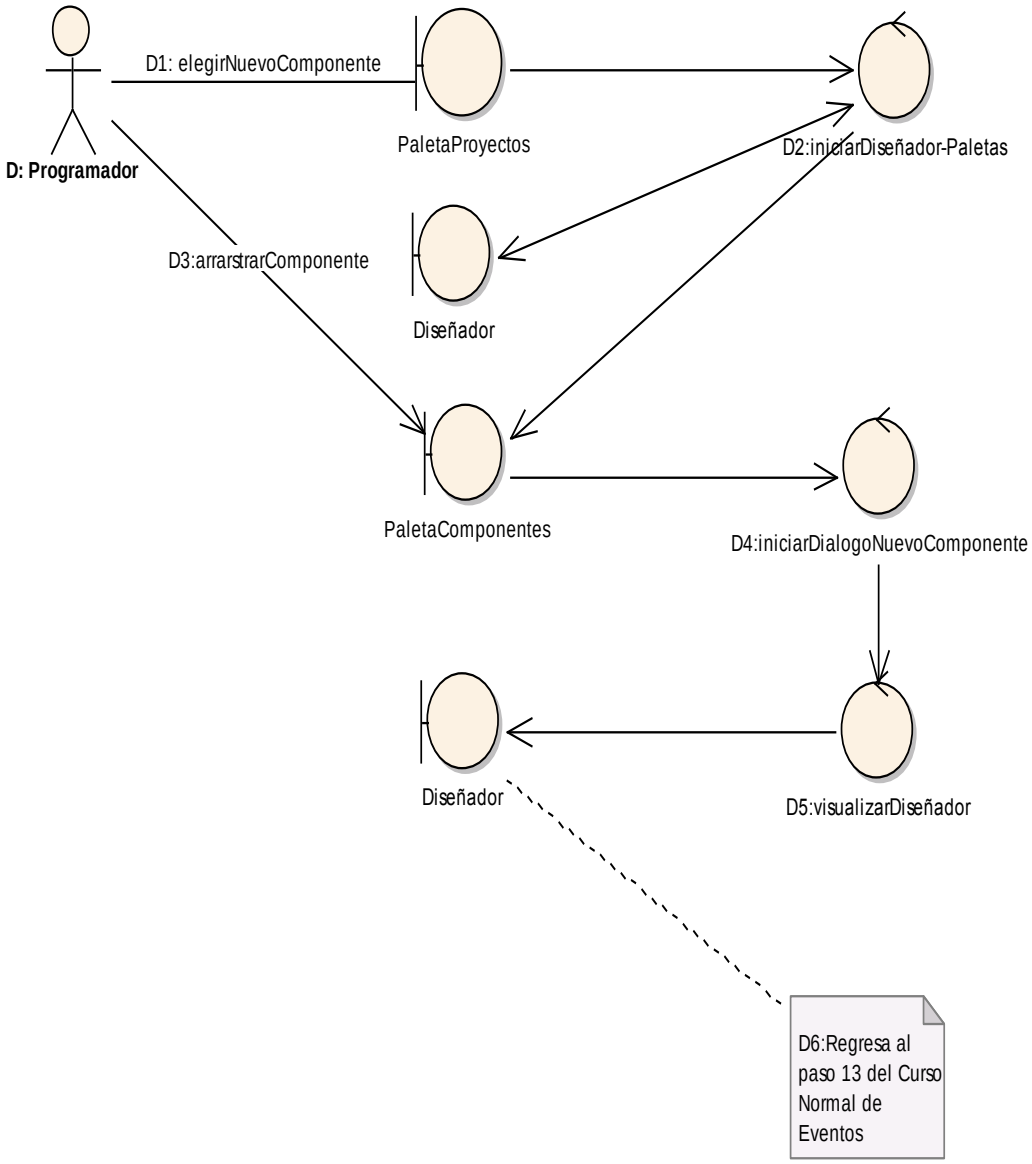



Figura 10.- Curso Alterno D: Nuevo Componente.

## CURSO ALTERNO E: CONSTRUIR UN PROYECTO "BUILD"

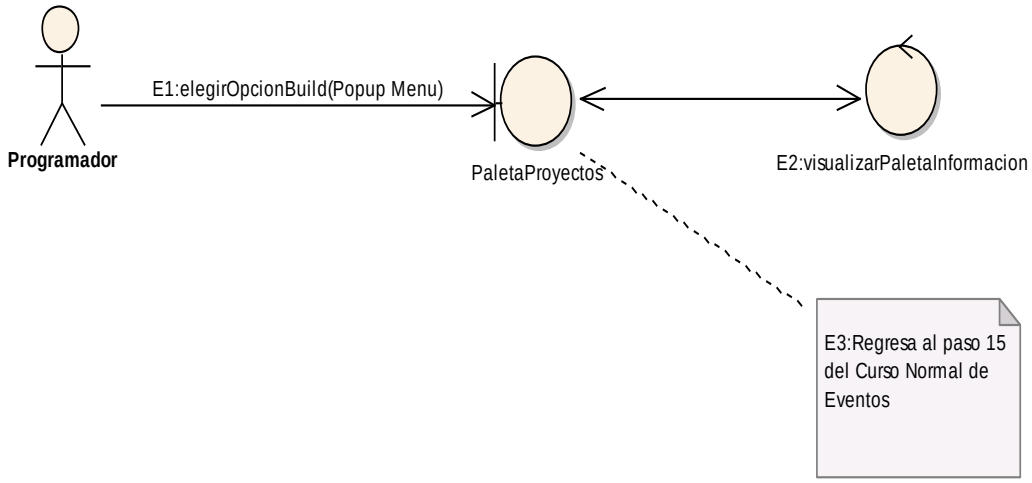



Figura 11.- Curso Alterno E: Construir Proyecto (Build).

## CURSO ALTERNO F: LIMPIAR PROYECTO "CLEAN"

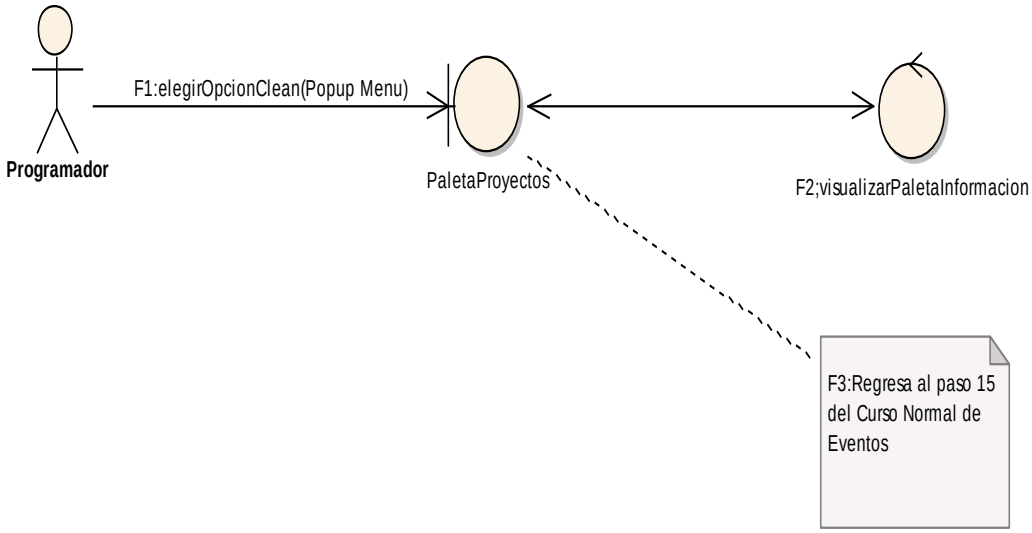



Figura 12.- Curso Alterno E: Limpiar Proyecto (Clean).

## DIAGRAMA DE SECUENCIA: ADMINISTRAR PROYECTO (Curso Normal)

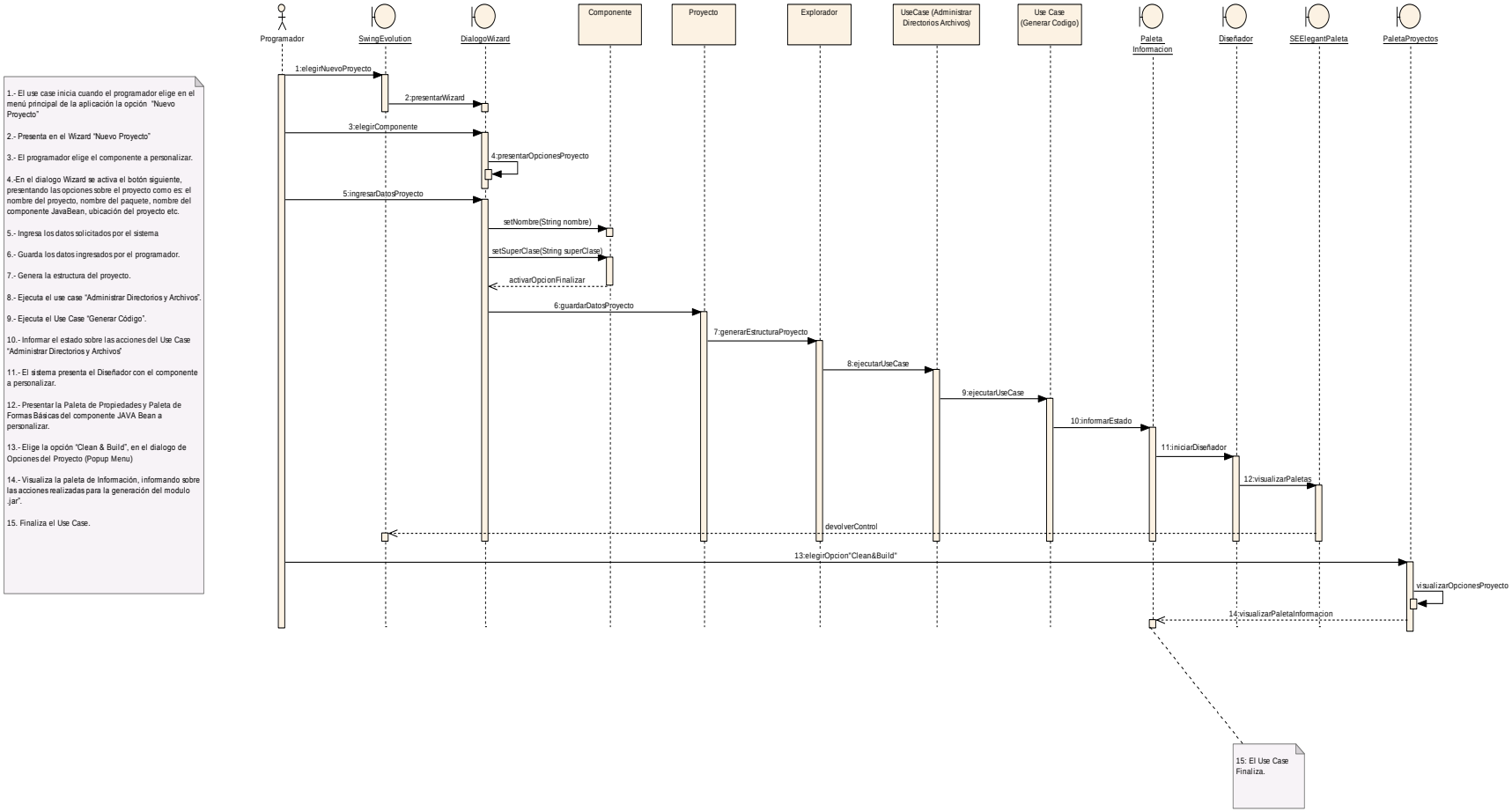



Figura 13.- Diagrama Secuencia Administrar Proyecto.

CURSO ALTERNO A: ABRIR PROYECTO

Viene del paso 1 del Curso Normal de Eventos

A1.- En el paso1 del Curso Normal de eventos el programador elige la opción de Abrir Proyecto.

A2.- El sistema presenta un cuadro de dialogo donde se debe elegir la ubicación del proyecto a abrir

A3.- Selecciona el proyecto y lo abre.

A4.- Abre el proyecto seleccionado por el programador.

A5.- El sistema presenta todo el proyecto en la paleta de proyectos, para continuar con la creación de nuevos componentes.

A6.- Reconstruye el árbol de Directorios del proyecto que se desea abrir.

A7.- Regresa al paso 17 del Curso Normal de Eventos.

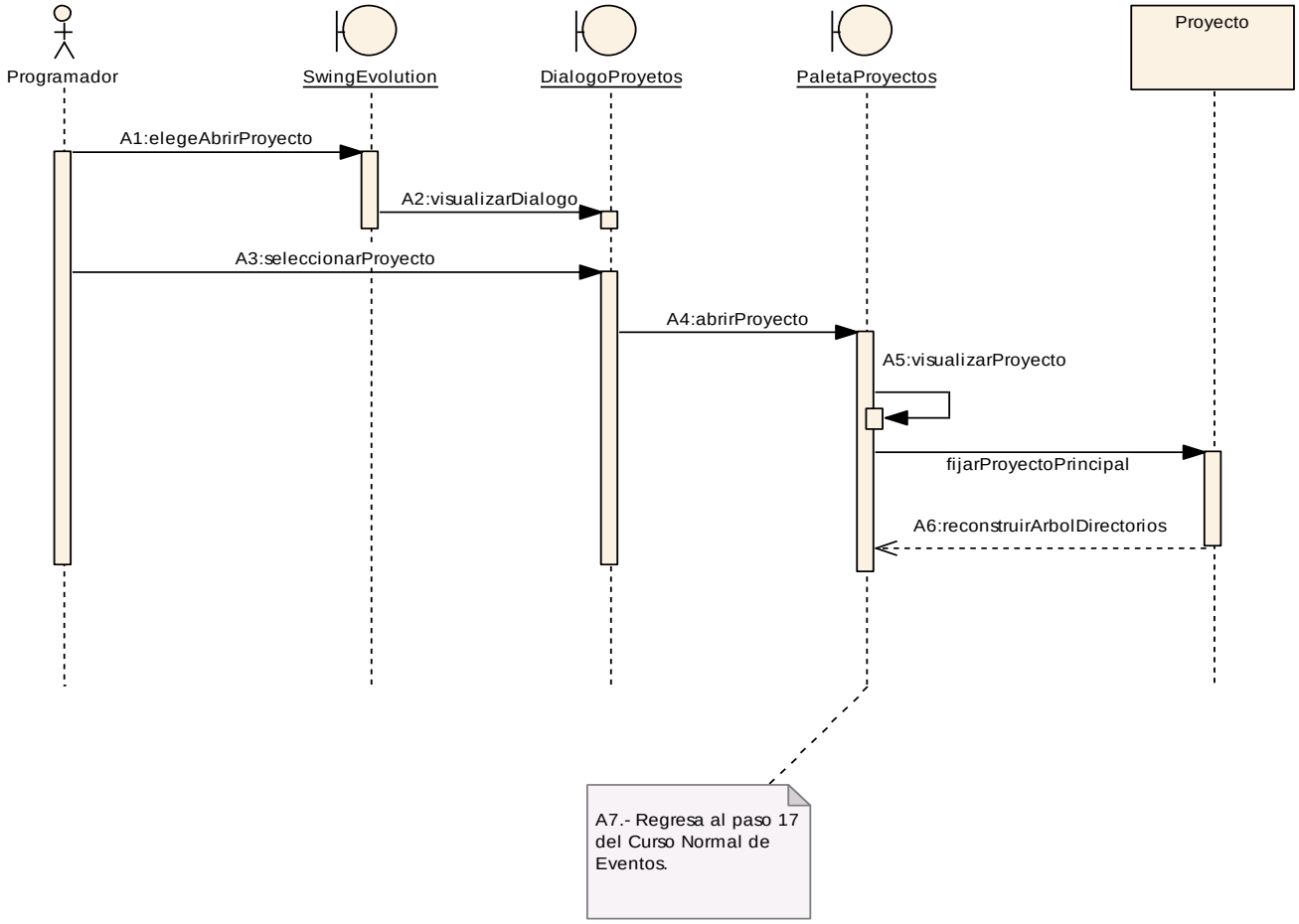


Figura 14.- Diagrama Secuencia Curso Alterno A: Abrir Proyecto.

CURSO ALTERNO B: ELIMINAR PROYECTO

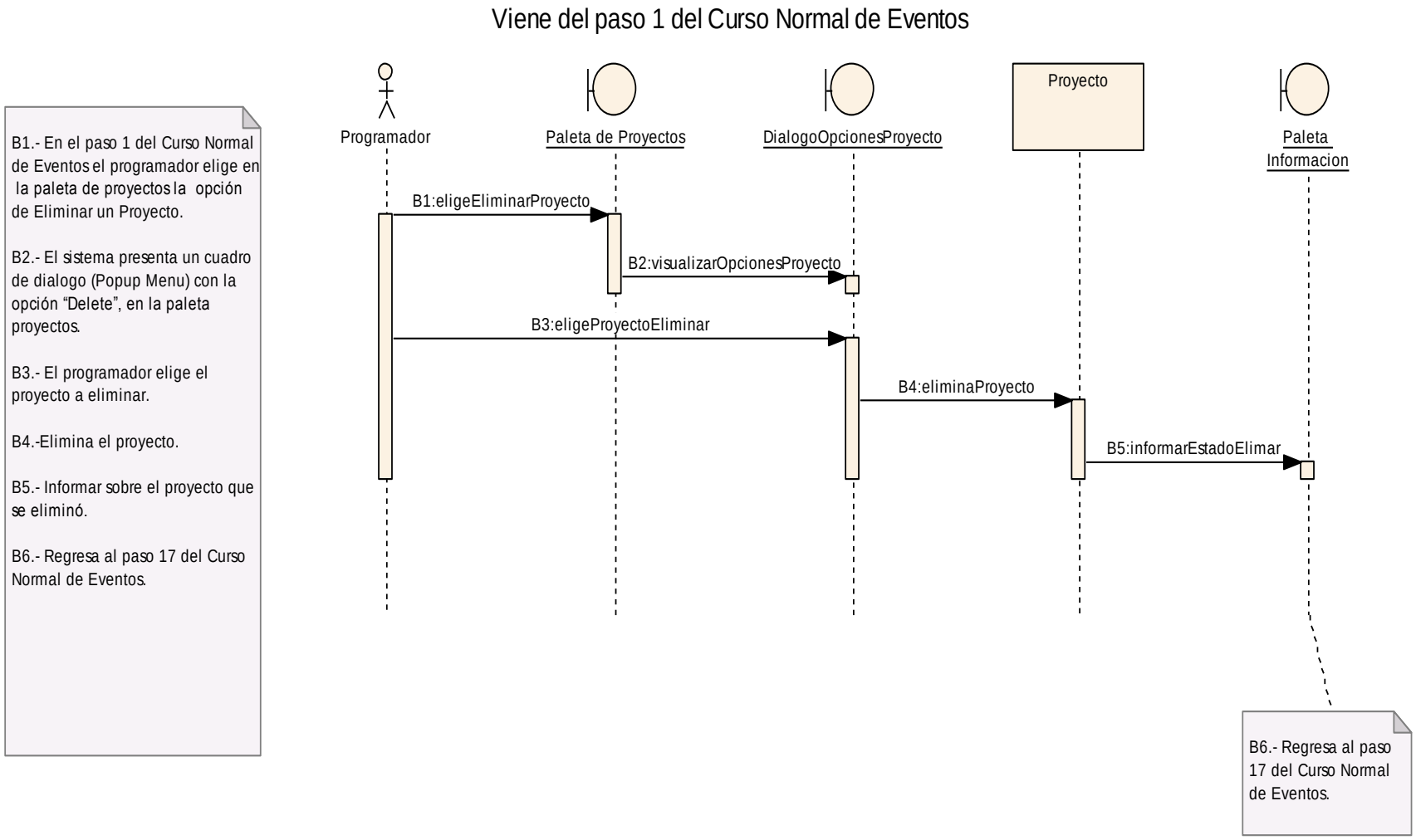



Figura 15.- Diagrama Secuencia Curso Alterno B: Eliminar Proyecto.

CURSO ALTERNO C: CERRAR PROYECTO

C: Viene del paso 1 del Curso Normal de Eventos

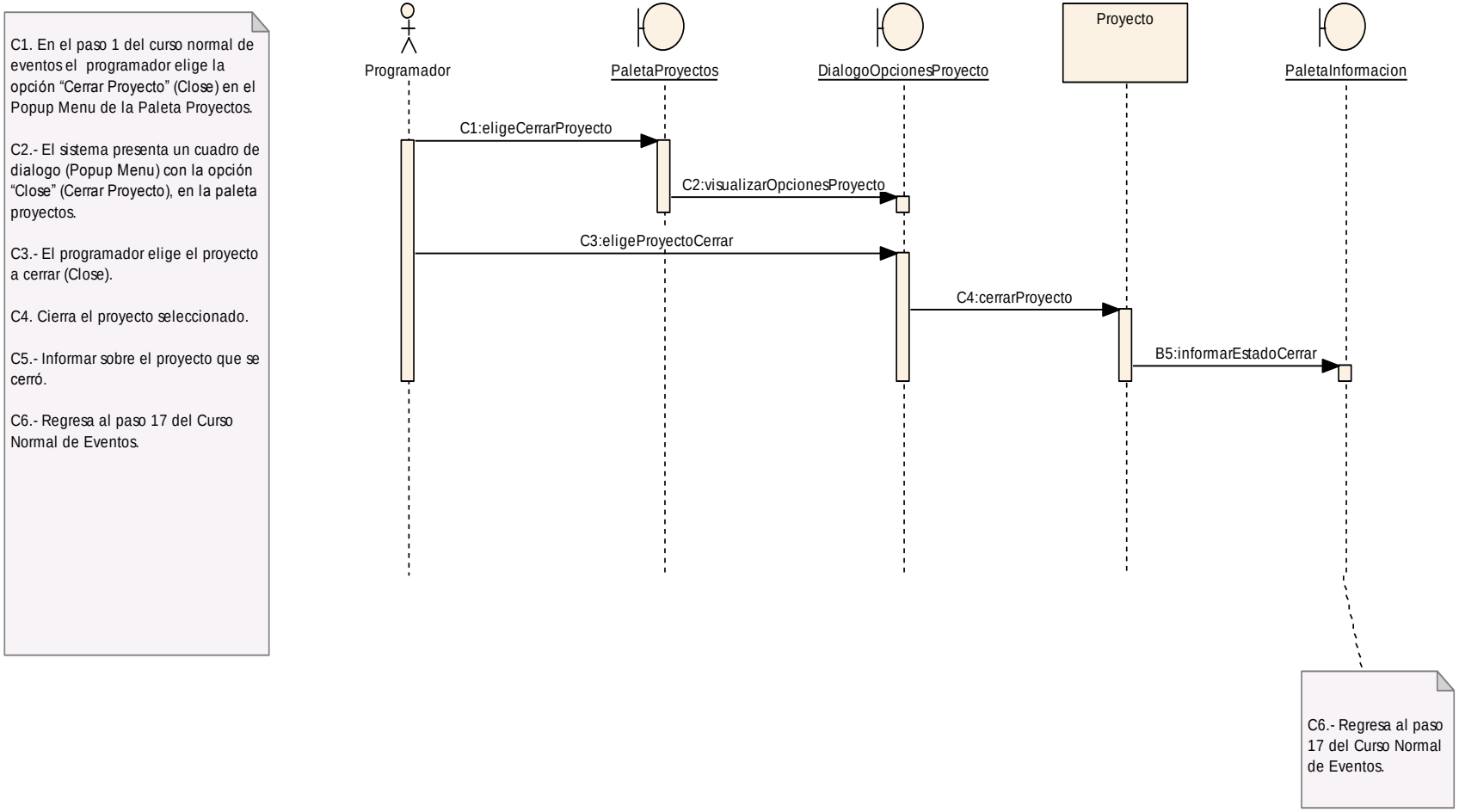



Figura 16.- Diagrama Secuencia Curso Alterno C: Cerrar Proyecto

CURSO ALTERNO D: NUEVO COMPONENTE

Viene del paso 13 del Curso Normal de Eventos

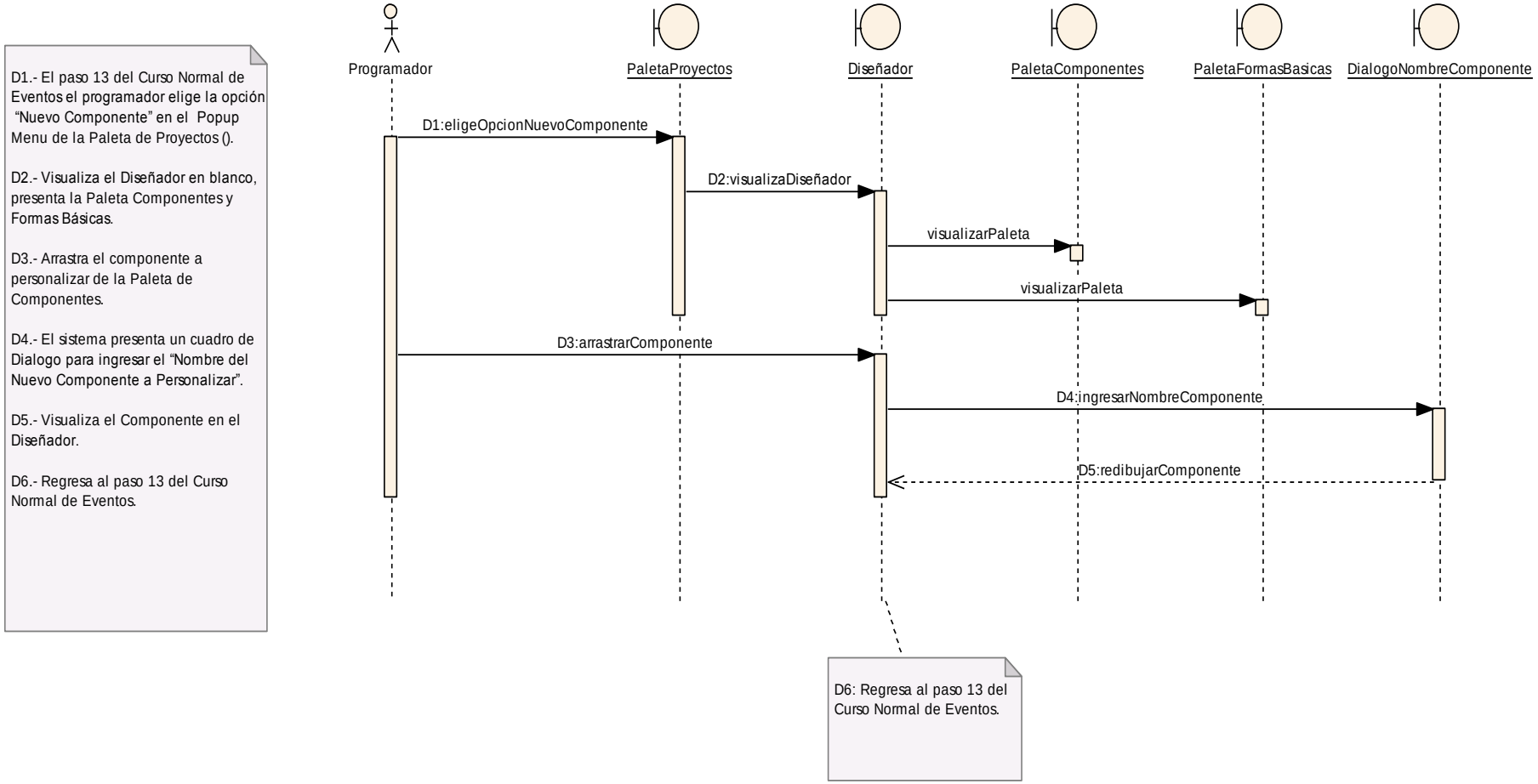



Figura 17.- Diagrama Secuencia Curso Alterno D: Nuevo Componente.

CURSO ALTERNO E: CONSTRUIR PROYECTO (BUILD)

**VIENE DEL PASO 13 DEL CURSO NORMAL DE EVENTOS**

E1.- El paso 13 del Curso Normal de Eventos el programador elige la opción "Build" en el Popup Menu de la Paleta de Proyectos.

E2.- Visualiza la paleta de Información, informando sobre la realización de la Tarea "Build" seleccionada por el programador.

E3.- Regresa al paso 15 del Curso Normal de Eventos.

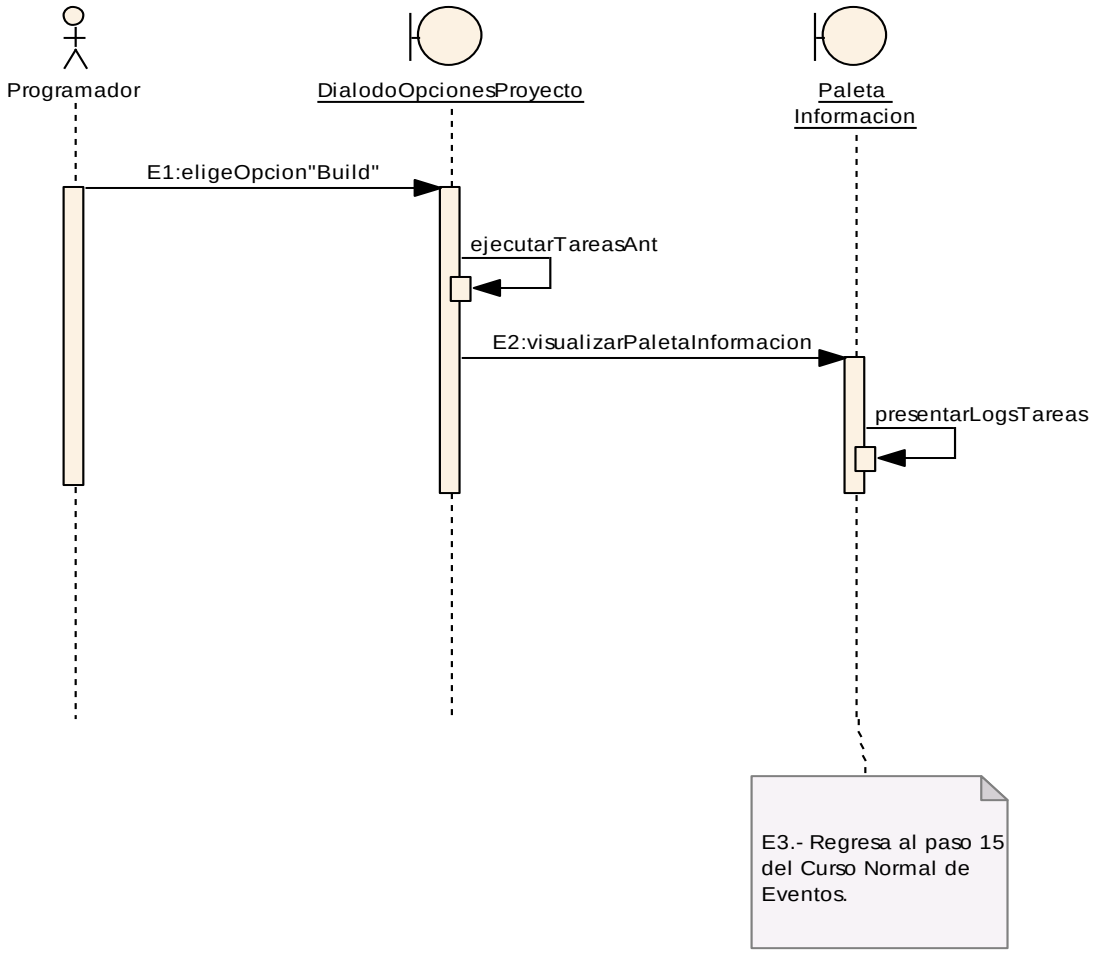


Figura 18.- Diagrama Secuencia Curso Alterno E: Construir Proyecto (Build &Clean).

CURSO ALTERNO F: LIMPIAR PROYECTO (CLEAN)

**VIENE DEL PASO 13 DEL CURSO NORMAL DE EVENTOS**

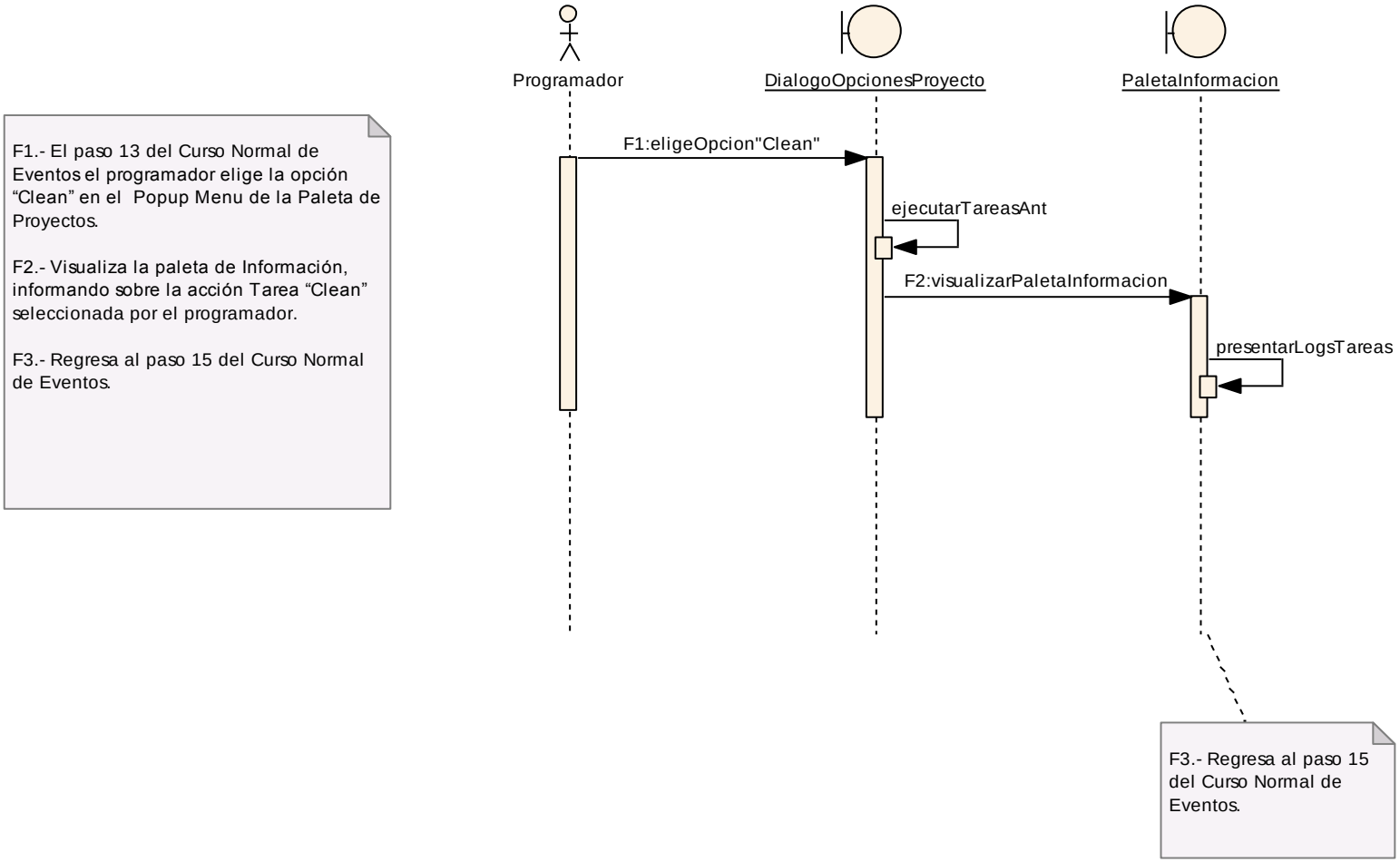



Figura 19.- Diagrama Secuencia Curso Alterno E: Limpiar Proyecto (Clean)

#### 6.3.2.2. Use Case: Administrar Plantilla.

| | |
|---|---|
| **Nombre:** | **Administrar Plantilla** |
| **Actor(es):** | Programador |
| **Propósito:** | Permitir Crear, abrir, guardar, eliminar, deshacer y rehacer una plantilla. |
| **Visión General** | Proporcionar al programador una forma fácil de trabajar con las plantillas, como es la creación, modificación y eliminación. |
| **Tipo:** | Primario, esencial |
| **Referencias:** | RF09. |
| **Precondición(es)** | |
| **Post-condiciones(es)** | |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Programador** | **Sistema** |
| 1. El use case inicia cuando el programador elige en el menú principal de la aplicación "Nueva Plantilla" | 2.- Presenta un Wizard (Dialogo) para la creación de una Nueva Plantilla. |
| 3. El programador elige la opción de plantilla vacía. | 4.- Inicia el dialogo "Nueva Plantilla" para recolectar datos referentes a la plantilla a crear como por ejemplo: nombre, tipo de plantilla, etc. |
| 5.- Ingresa los datos correspondientes de la plantilla a crear. | 6.- Guarda los datos ingresados. |
| | 7.- Presenta (Activa) la paleta de formas Básicas para la creación de la plantilla. |
| | 8.- Informa estado sobre la generación de código XML de la plantilla. |
| | 9.- Ejecuta el Use Case Modificar Apariencia. |

| | |
|---|---|
| 12.- El programador elige la opción de deshacer los avances realizados en la plantilla.<br><br>17.- El programador elige guardar la plantilla. | 10.- Informar cambios a causa del Use Case "Modificar Apariencia".<br><br>11.- Presenta las opciones de deshacer y rehacer los avances de la plantilla.<br><br>13.- Consulta el estado pasado para restablecerlo.<br><br>14.- Redibuja en el diseñador el estado anterior de la plantilla<br><br>15.- Informa sobre la generación de código (XML) de la plantilla.<br><br>16.- Presenta la opción de "Aceptar Figura" creada para la plantilla.<br><br>18.- Guarda la plantilla.<br><br>19.- Finaliza el use case. |
| **CURSO ALTERNO DE EVENTOS** | |
| **A.- ABRIR PLANTILLA** | |
| A1.- En el paso 1 del Curso Normal de eventos el programador elige la opción de Abrir Plantilla.<br><br>A3.- Selecciona la plantilla y la abre. | A2.- El sistema presenta un cuadro de dialogo donde elegir la ubicación de las plantillas guardadas.<br><br>A4. Abre la plantilla seleccionada por el programador.<br><br>A5.- Redibuja toda la plantilla en el diseñador.<br><br>A6.- Regresa al paso 7 del Curso Normal de Eventos. |

| | |
|---|---|
| **B.- ELIMINAR PLANTILLA** | |
| B1.- En el paso1 del Curso Normal de Eventos el programador elige la opción de Eliminar Plantilla en la paleta de plantillas.<br><br>B3.- El programador elige la plantilla a eliminar en la paleta plantillas. | B2.- El sistema presenta la paleta de plantillas creadas en la herramienta.<br><br>B4.- Borra la plantilla seleccionada por el programador.<br><br>B5.- El sistema regresa al paso 19 del Curso Normal de Eventos. |
| **C.- REHACER PLANTILLA** | |
| C1.- En el paso 12 del curso normal de eventos el programador elige la opción rehacer. | C2.-El sistema restablece la plantilla según los avances realizados y almacenados en el historial.<br><br>C3. Redibuja la plantilla en el diseñador.<br><br>C4. Regresa el paso 15 del Curso Normal de Eventos. |

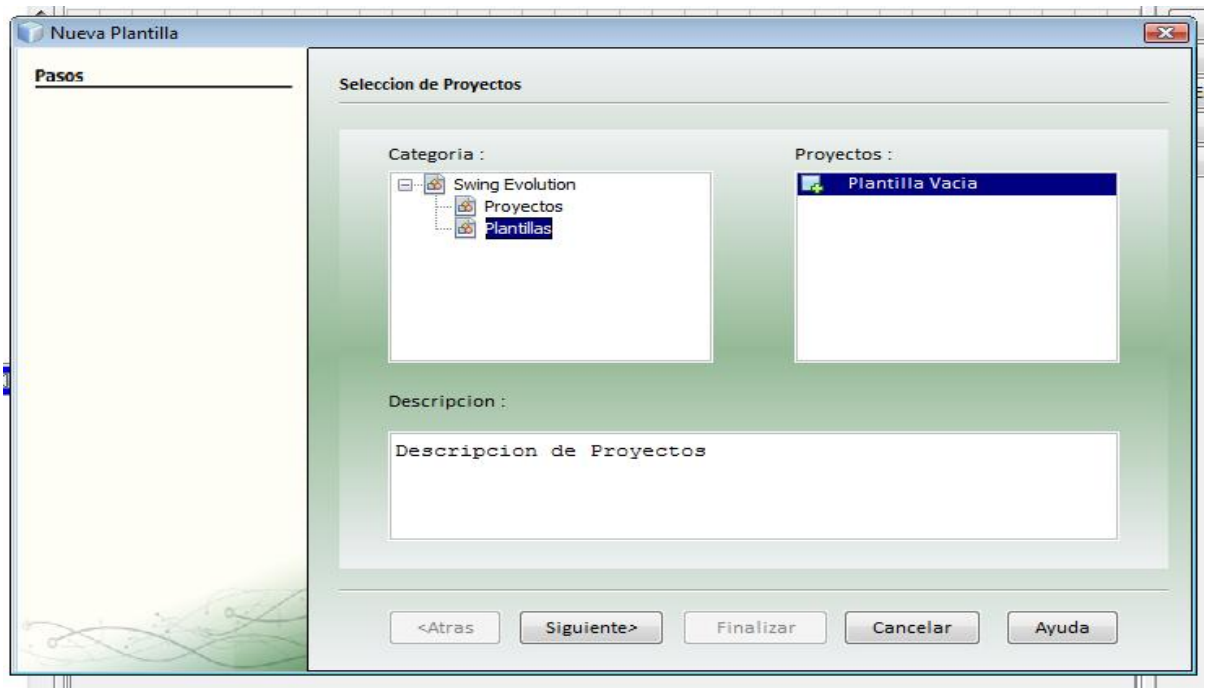



**Figura 20. Pantalla Administrar Plantilla (Nuevo Plantilla).**

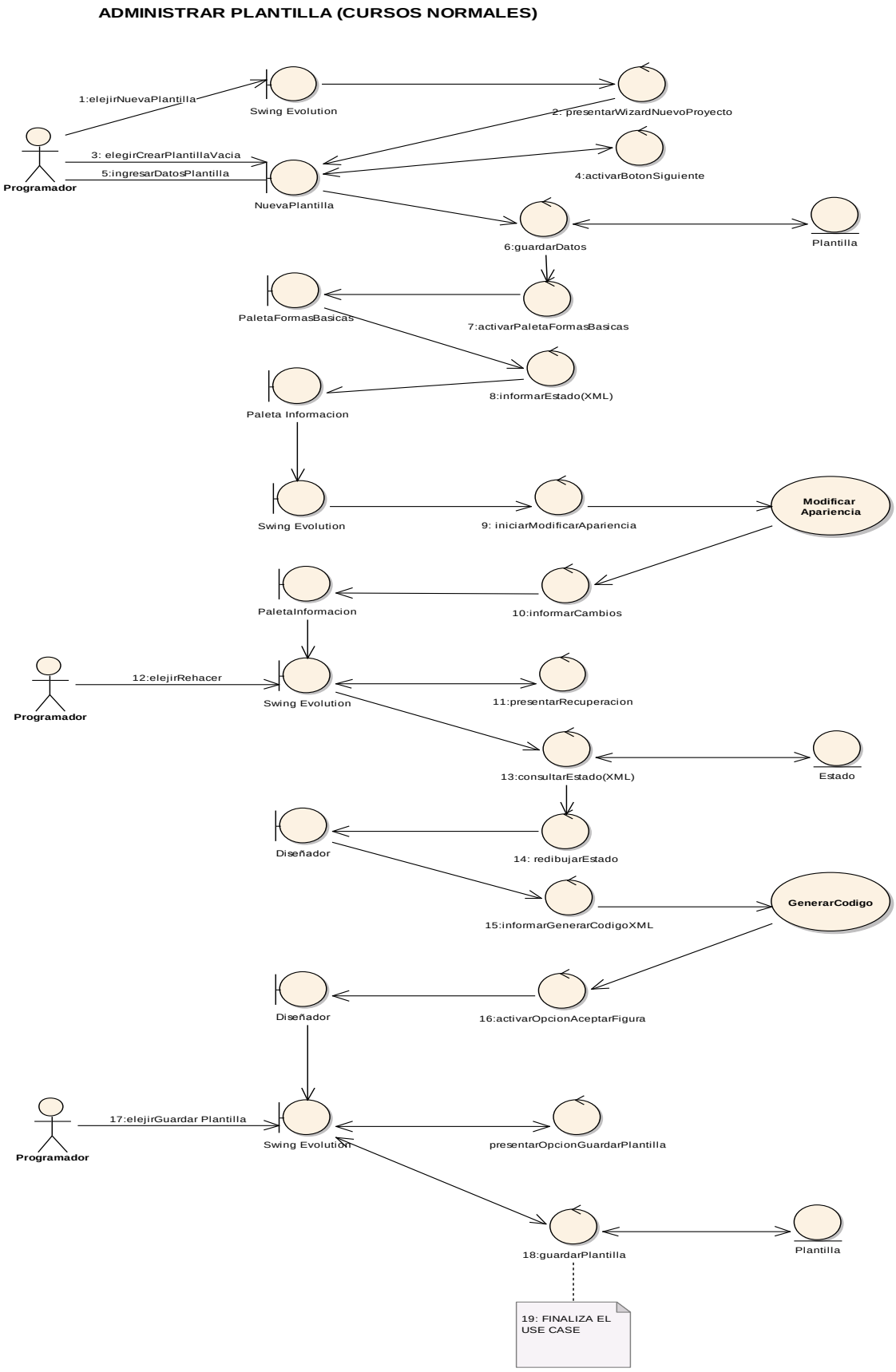



Figura 21.- Curso Normal Use Case: Administrar Plantilla.

## ADMINISTRAR PLANTILLA (CURSOS ALTERNOS)

Viene del paso 1 del Curso Normal de Eventos

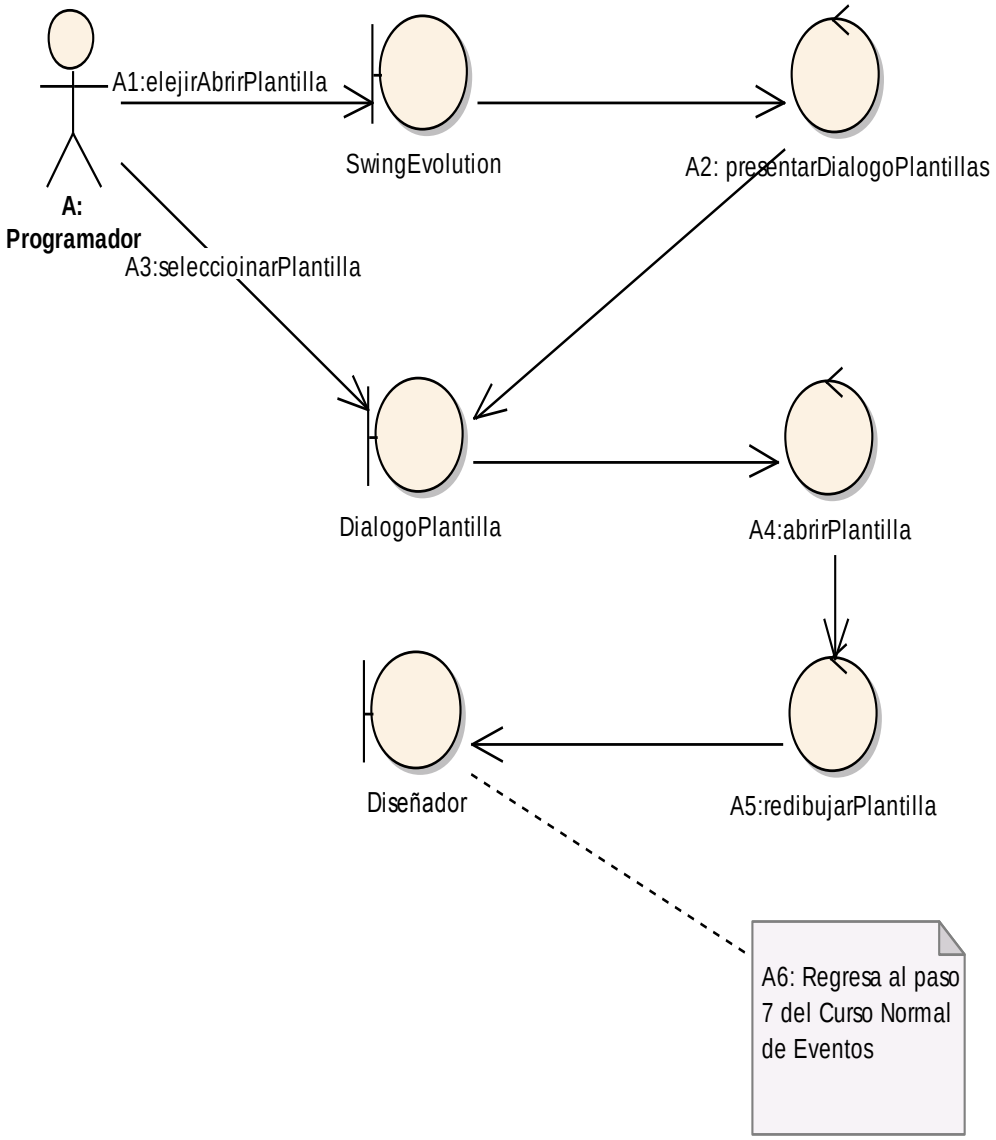



Figura 22.- Curso Alterno A: Abrir Plantilla.

## CURSO ALTERNO B : ELIMINAR PLANTILLA

Viene del paso 1 del Curso Normal de Eventos

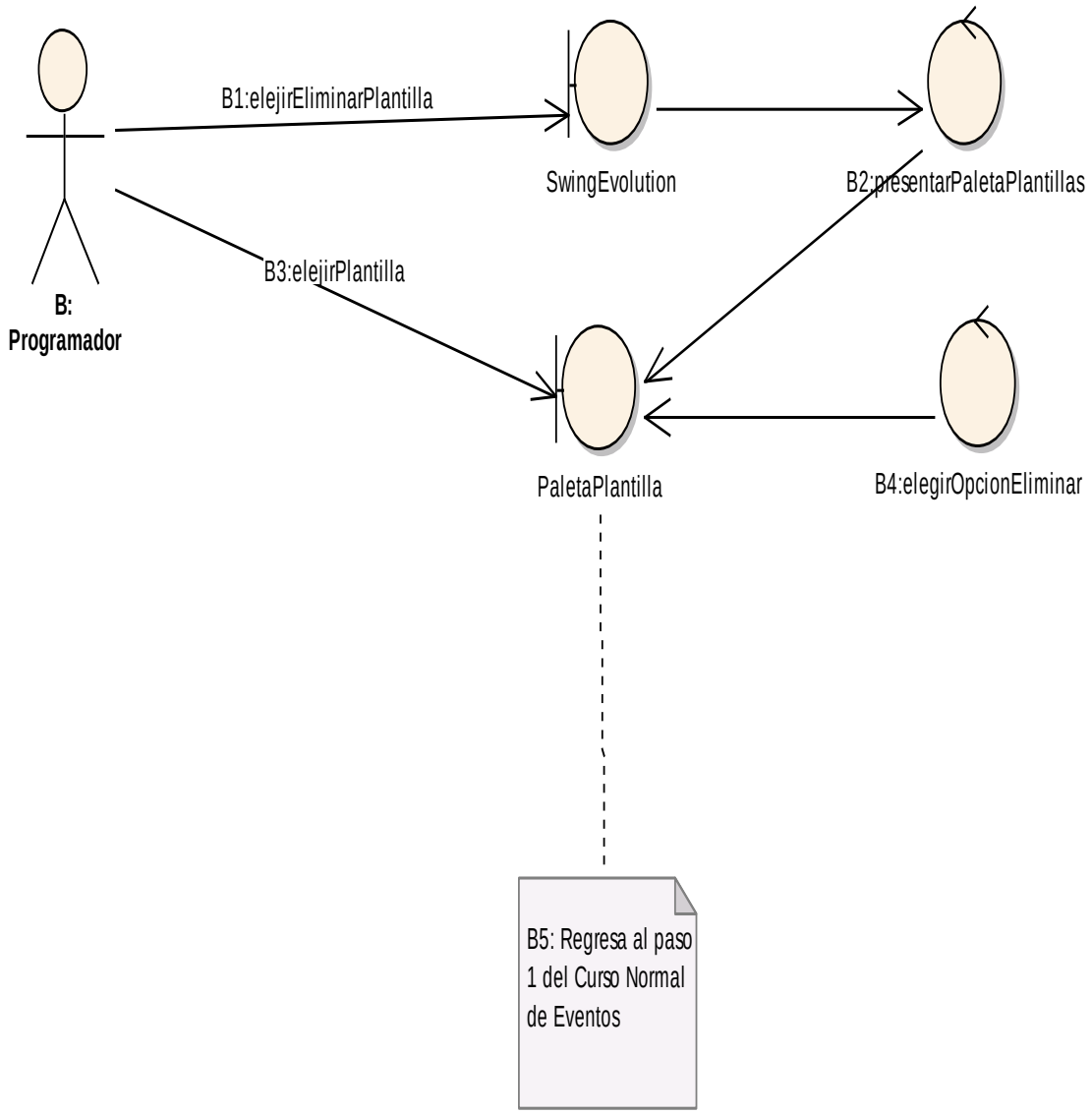



Figura 23.- Curso Alterno B: Eliminar Plantilla.

# CURSO ALTERNO C: REHACER PLANTILLA

Viene del paso 12 del Curso Normal de Eventos

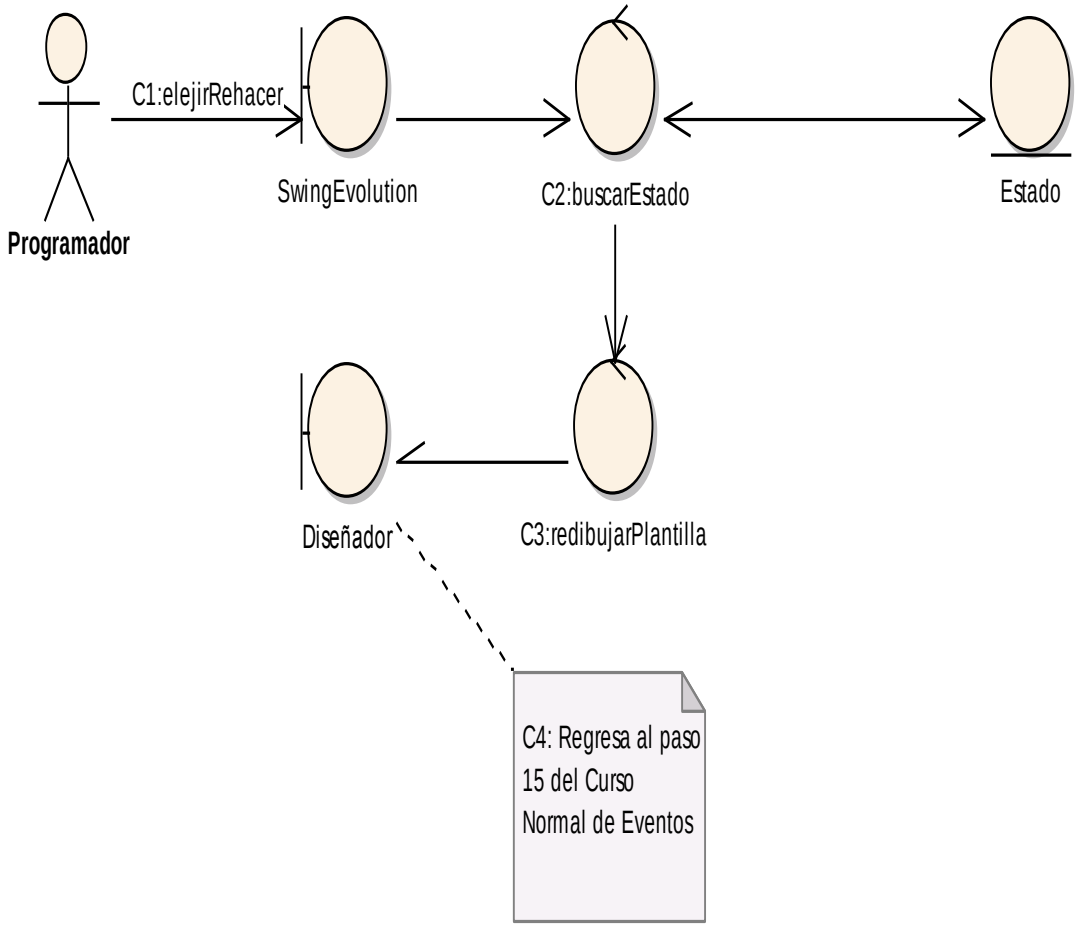



Figura 24.- Curso Alterno C: Rehacer Plantilla.

DIAGRAMA DE SECUENCIA: ADMINISTRAR PLANTILLA (Curso Normal)

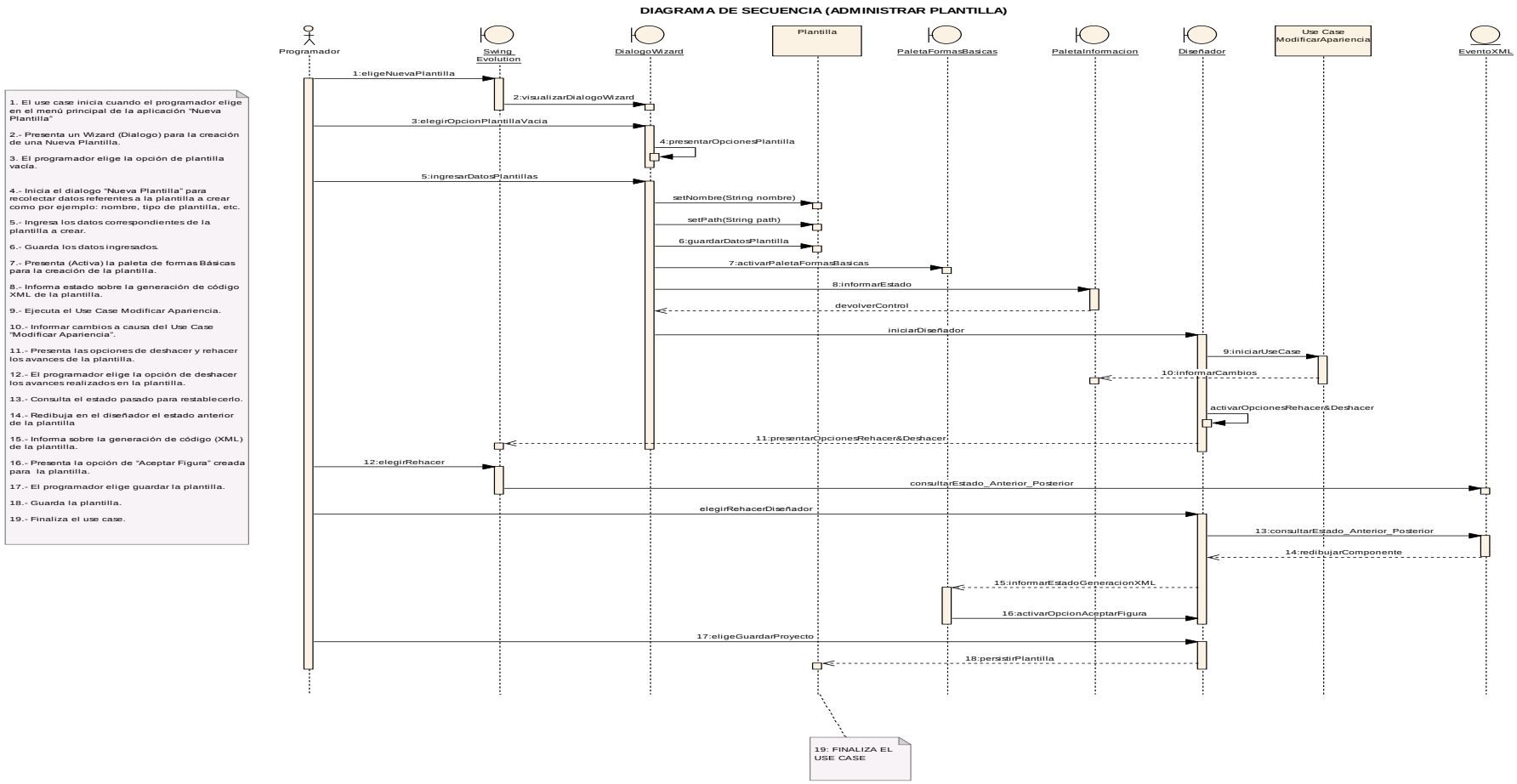



Figura 25.- Diagrama Secuencia Administrar Plantilla.

CURSO ALTERNO A: ABRIR PLANTILLA

**Viene del paso 1 del Curso Normal de Eventos**

A1.- En el paso 1 del Curso Normal de eventos el programador elige la opción de Abrir Plantilla.

A2.- El sistema presenta un cuadro de dialogo donde elegir la ubicación de las plantillas guardadas.

A3.- Selecciona la plantilla y la abre.

A4. Abre la plantilla seleccionada por el programador.

A5.- Redibuja toda la plantilla en el diseñador.

A6.- Regresa al paso 7 del Curso Normal de Eventos.

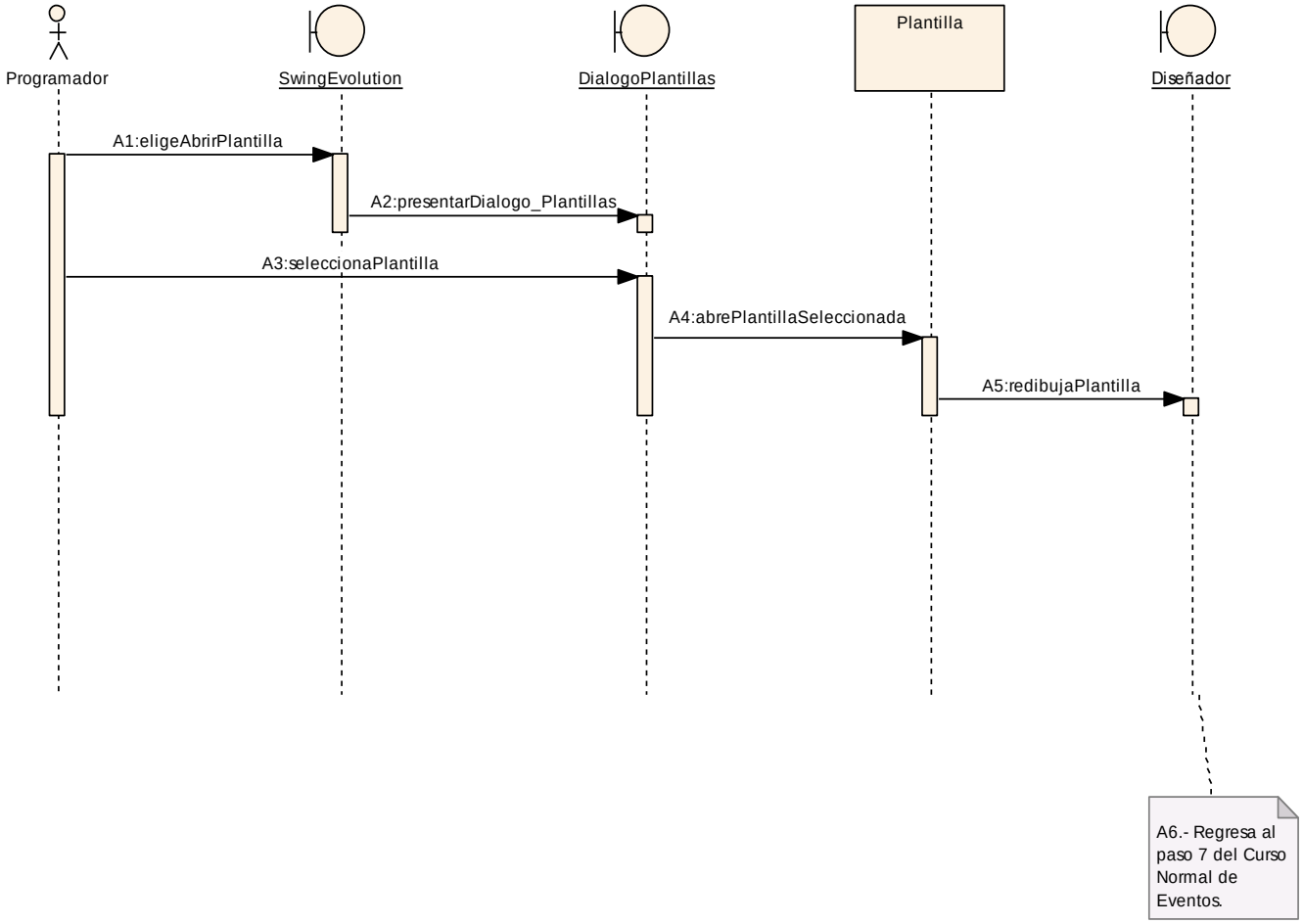


Figura 26.- Diagrama Secuencia Curso Alterno A: Abrir Plantilla.

CURSO ALTERNO B: ELIMINAR PLANTILLA

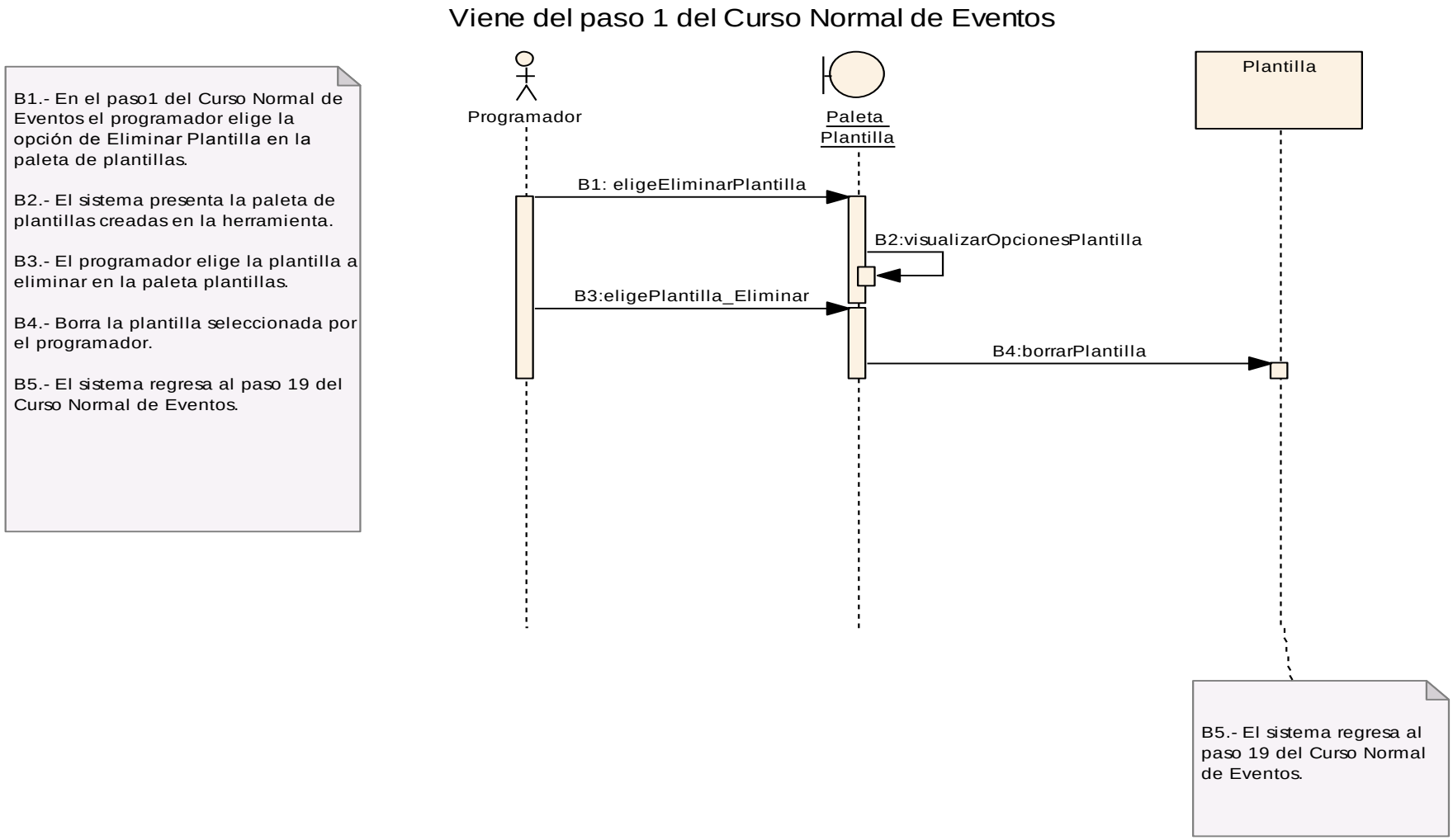



Figura 27.- Diagrama Secuencia Curso Alterno B: Eliminar Plantilla

CURSO ALTERNO C: REHACER PLANTILLA

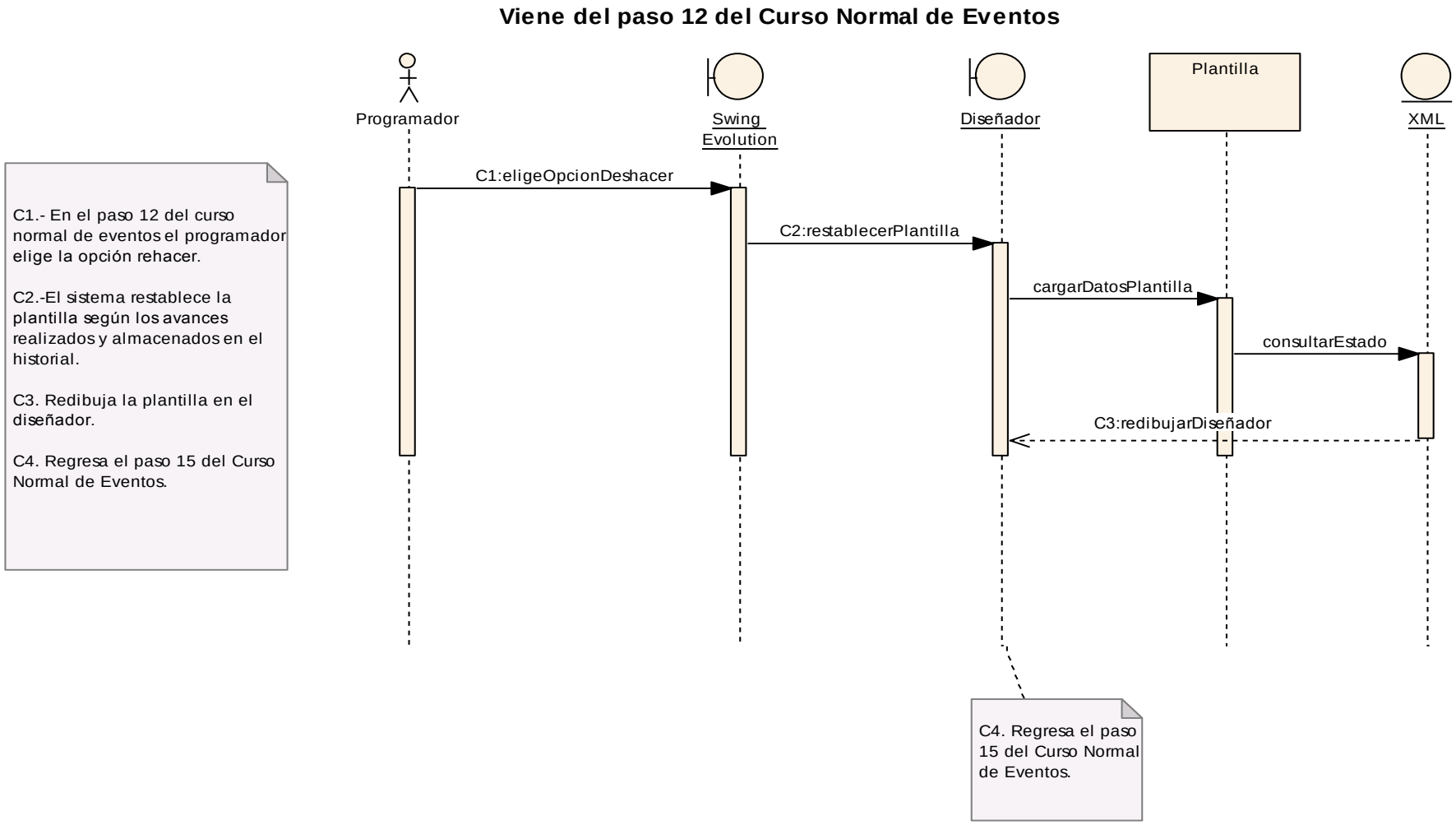



Figura 28.- Diagrama Secuencia  Curso Alterno C: Rehacer Plantilla.

#### 6.3.2.3. Use Case: Modificar Apariencia.

| | |
|---|---|
| **Nombre:** | **Modificar Apariencia** |
| **Actor(es):** | Programador. |
| **Propósito:** | Proporcionar las acciones pertinentes para dar nuevas formas a los JButton o JToogleButton. |
| **Visión General** | Obtener características de los nuevos componentes y permitir visualizarlos en el diseñador |
| **Tipo:** | Primario, esencial. |
| **Referencias:** | RF01, RF11. |
| **Precondición(es)** | Que el componente a modificar sea un JButton o JToogleButton además haber ejecutado cualquier Use Case ya sea "Administrar Proyecto" o "Administrar Plantilla" |
| **Post-condiciones(es)** | |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Programador** | **Sistema** |
| 1.- El use case inicia cuando el programador arrastra la figura deseada para dar nueva forma al componente que está en el diseñador | 2.- Reemplaza el componente que se encuentra en el diseñador por la figura arrastrada. |
| | 3.- Activa los controles de "Paleta de Combinaciones" para la manipulación de las figuras. |
| 4.- Elije opción "aceptar figura" del campo "Aceptación Final" de la "Paleta de Combinaciones". | 5.- Reemplaza la figura por el componente que estamos personalizando pero ahora con la figura como nueva apariencia del componente. |
| | 6.- Inicia el Use Case "Generar Código". |
| | 7.- Informa sobre la generación de código. |

| | |
|---|---|
| | 8.- Redibuja el componente con todas las modificaciones realizadas.<br><br>9.-finaliza el use case. |
| **CURSO ALTERNO DE EVENTOS** | |
| **A.- ARRASTRAR FIGURA** | |
| A1.- En el paso 4 del Curso Normal de Eventos el programador elije arrastrar una figura. | A2.- Agrega la figura arrastrada al diseñador.<br><br>A3.- Activa los controles del campo "OPERADORES" de la "Paleta Combinaciones".<br><br>A4.- Regresa al paso 4 del curso normal de eventos. |
| **B.- COMBINA DOS FIGURAS** | |
| B1.- En el paso 4 del Curso Normal de Eventos el programador elige combinar 2 figuras.<br><br>B2.- Elije las figuras a combinar en el campo "Operadores" de la "Paleta Combinaciones".<br><br>B3.- Del campo "Operadores" de la "Paleta Combinaciones", elije el operador se va a combinar las 2 figuras. | |

| | |
|---|---|
| B4.- Elije la opción “PROBAR OPERACION” de la “Paleta Combinaciones”. | |
| | B5.- Aplica el operador lógico escogido a las figuras pertinentes en el diseñador. |
| B7.- Elije la opción “Combinar” del campo “operadores” de la “Paleta Combinaciones”. | B6: Activa o desactiva controles de la “Paleta de Combinaciones” para la manipulación de las figuras. |
| B9.- Regresa al paso 4 del curso normal de eventos. | B8.- Aplica las operaciones pertinentes. |
| **C.- MOVER UNA FIGURA** | |
| C1.- En el paso 4 del curso normal de eventos el programador elige el número de la figura que quiere mover en el campo “Activación de figura” de la “Paleta Combinaciones”. | C2.- Cambia de color de la figura activada. |
| C3.- Se posiciona en el área de la figura activada. | |
| | C4.- Cambia el cursor que indique que está posicionado en el área de mover la figura. |
| C5.- Arrastra la figura al lugar deseado. | |
| | C6.- Redibuja la figura al nuevo lugar. |
| C7.-Regresa al paso 4 del curso normal de eventos. | |
| **D.- MODIFICAR EL TAMAÑO DE UNA FIGURA** | |
| D1.- En el paso 4 del curso normal de eventos el programador elige el número de la figura que quiere cambiar el tamaño en el | D2.- Cambia de color de la figura activada. |

| | |
|---|---|
| campo “Activación de figura” de la “Paleta Combinaciones”.<br><br>D3.- Se posiciona en los bordes de la figura activada.<br><br>D5.- Hace un arrastre de acuerdo al tamaño deseado.<br><br>D7.- Regresa al paso 4 del curso normal de eventos. | D4.- Cambia el cursor que indique que está posicionado en el área de modificar el tamaño de la figura.<br><br>D6.- Redibuja la figura con el nuevo tamaño. |

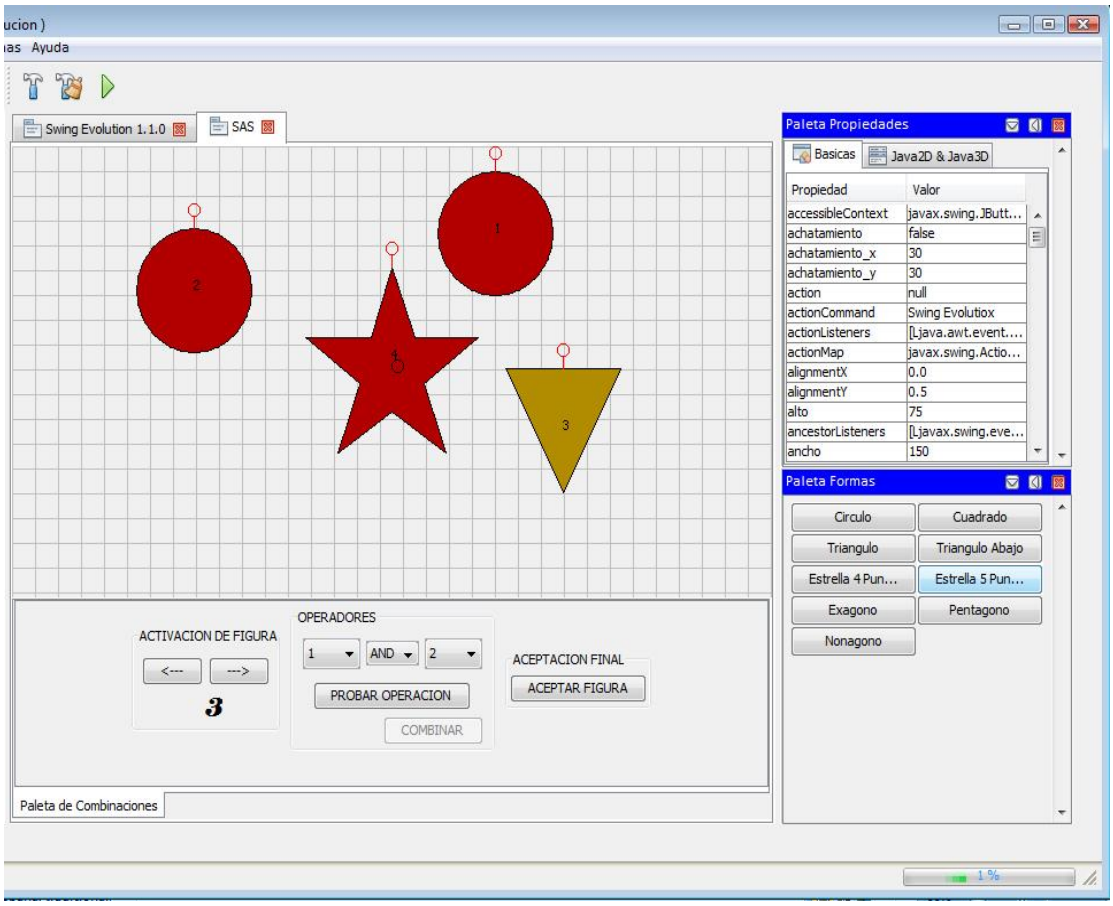



Figura 29. Modificar Apariencia (Arrastrar Figura).

## DIAGRAMA SECUENCIA: MODIFICAR APARIENCIA

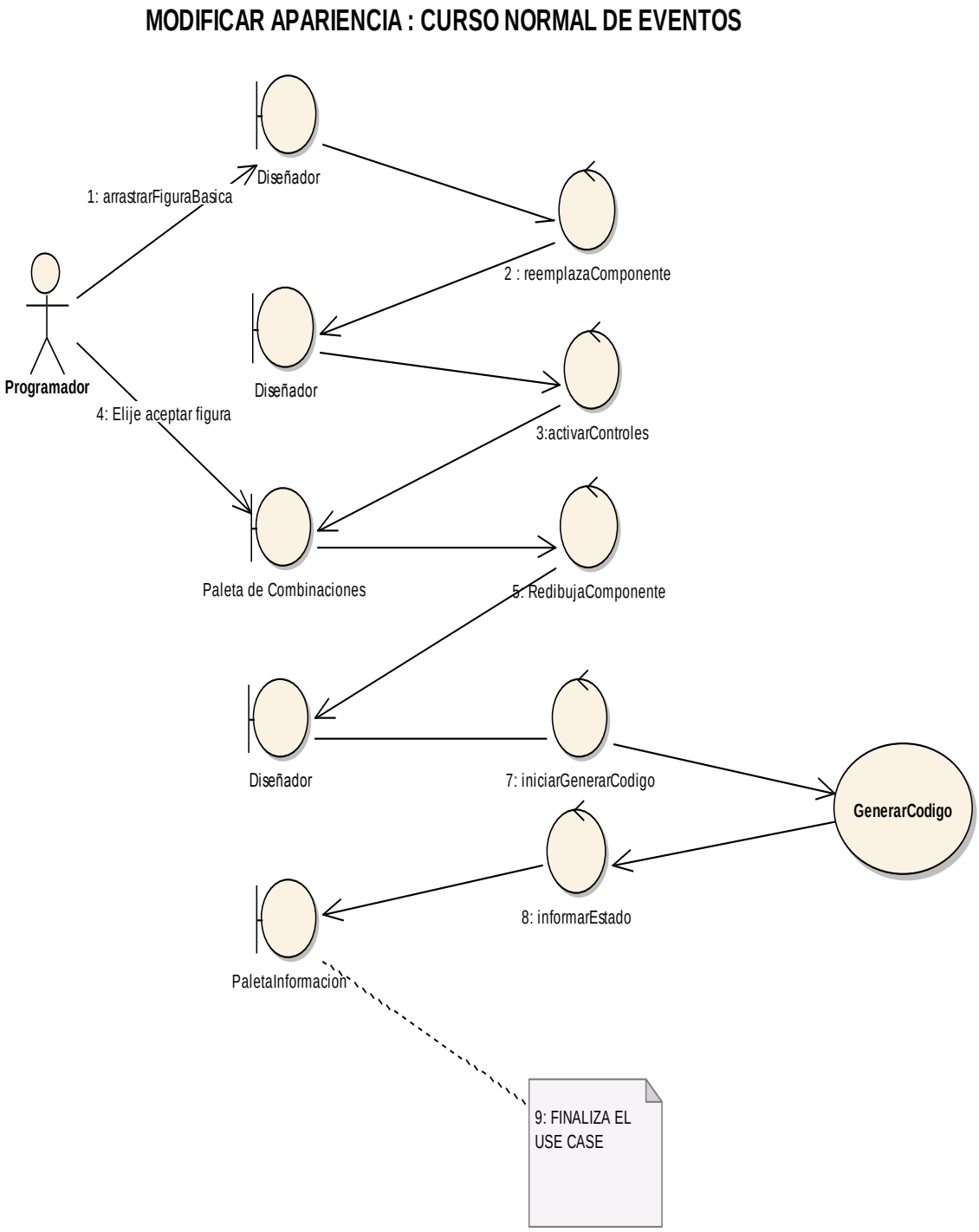



Figura 30.- Curso Normal: Modificar Apariencia.

# MODIFICAR APARIENCIA(CURSOS ALTERNOS)

## CURSO ALTERNO A: ARRASTRAR FIGURA

A: AGREGAR UNA FIGURA ADICIONAL (Viene del paso 4 del Curso Normal de Eventos)

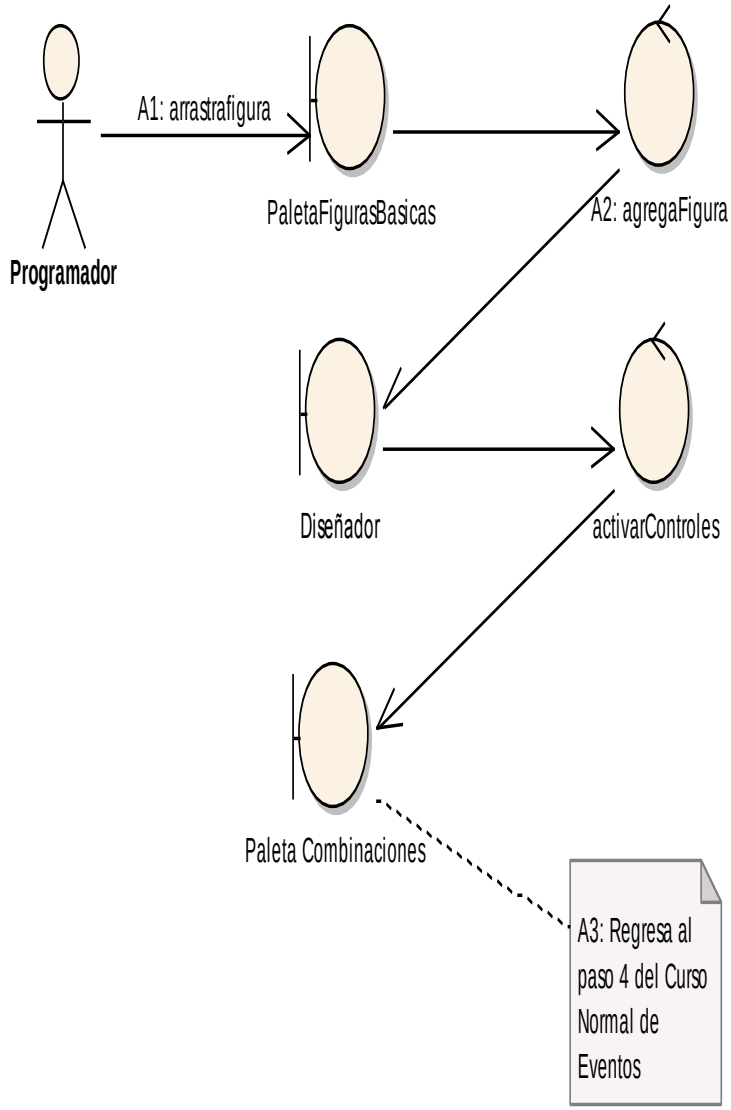



Figura 31.- Curso Alterno A: Arrastrar Figura.

## CURSO ALTERNO B: COMBINAR FIGURAS

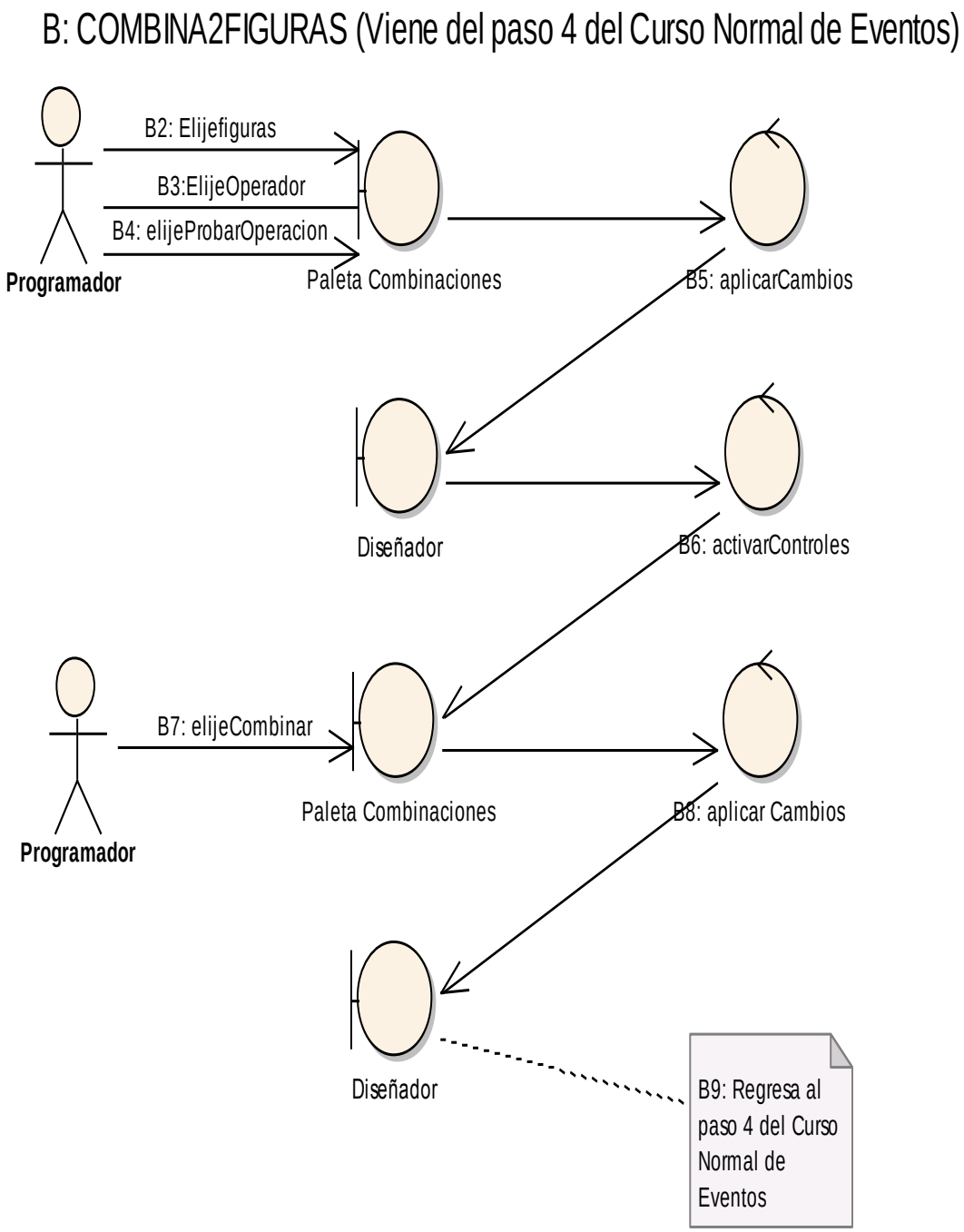



Figura 32.- Curso Alterno B: Combinar Figuras.

# CURSO ALTERNO C: MOVER FIGURAS

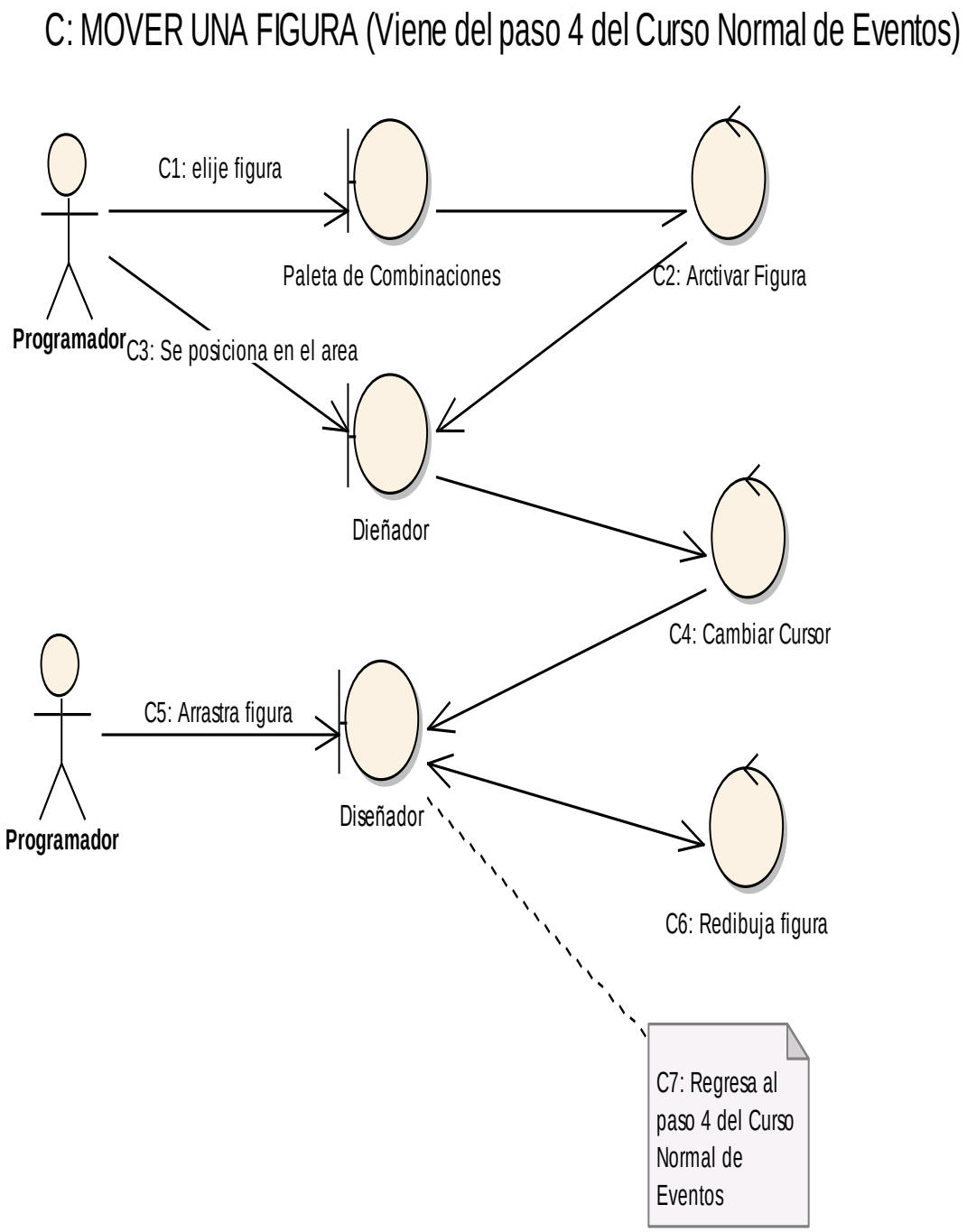



Figura 33.- Curso Alterno C: Mover Figuras.

# CURSO ALTERNO D: MODIFICAR TAMAÑO DE UNA FIGURA

## D: MODIFICAR EL TAMAÑO DE UNA FIGURA (Viene del paso 4 del Curso Normal de Eventos)

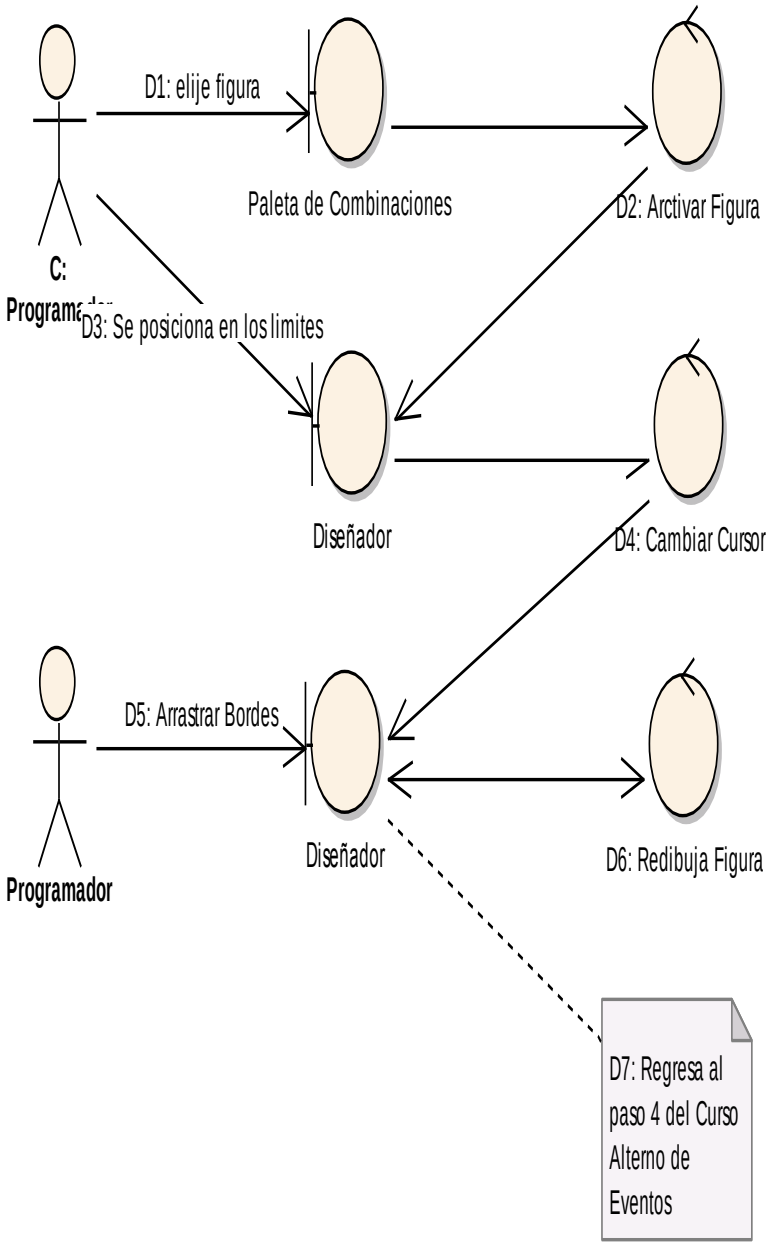



Figura 34.- Curso Alterno D: Modificar Tamaño de una Figura.

DIAGRAMA DE SECUENCIA: MODIFICAR APARIENCIA (Curso Normal)

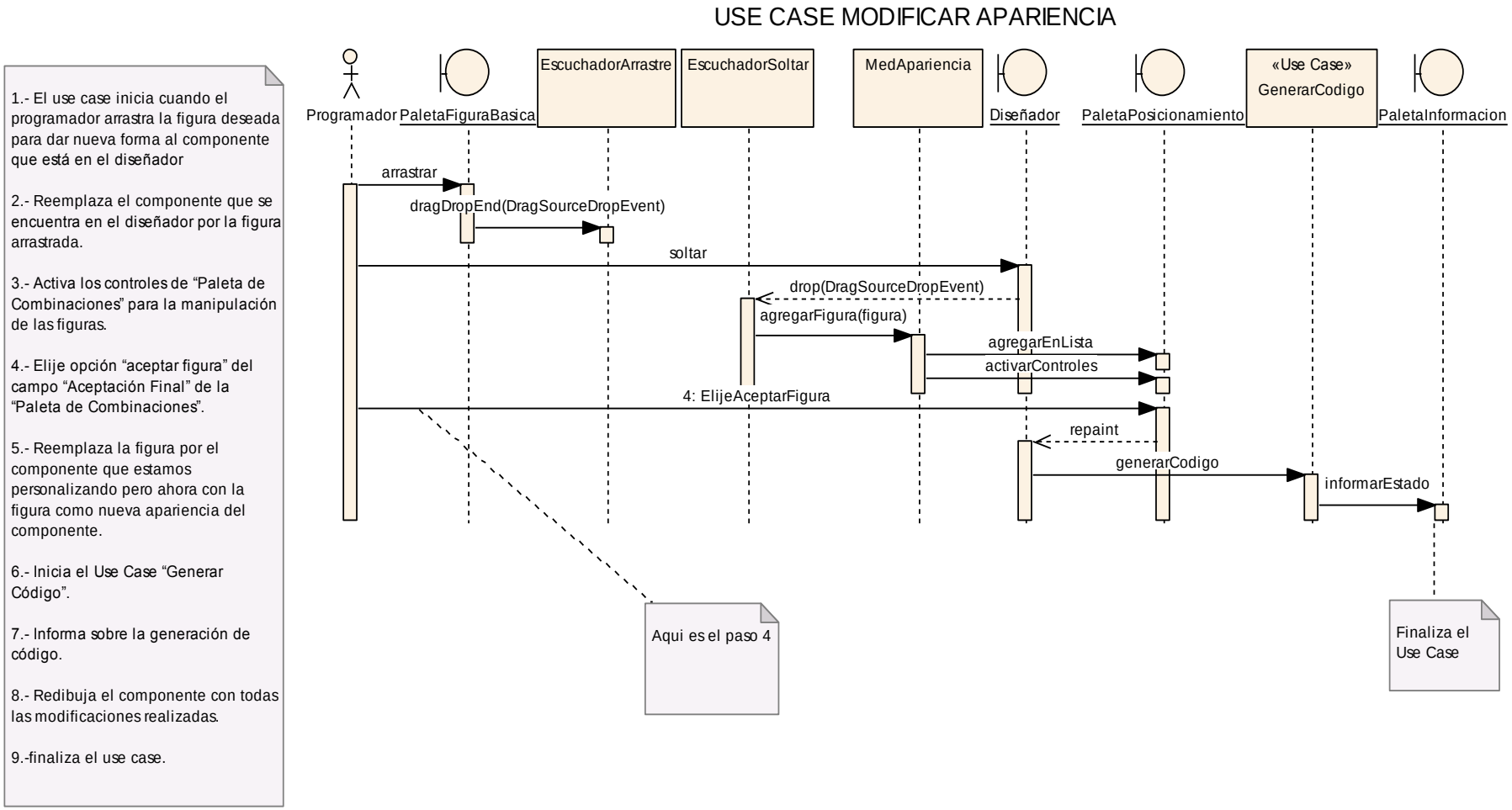



Figura 35.- Diagrama Secuencia Modificar Apariencia.

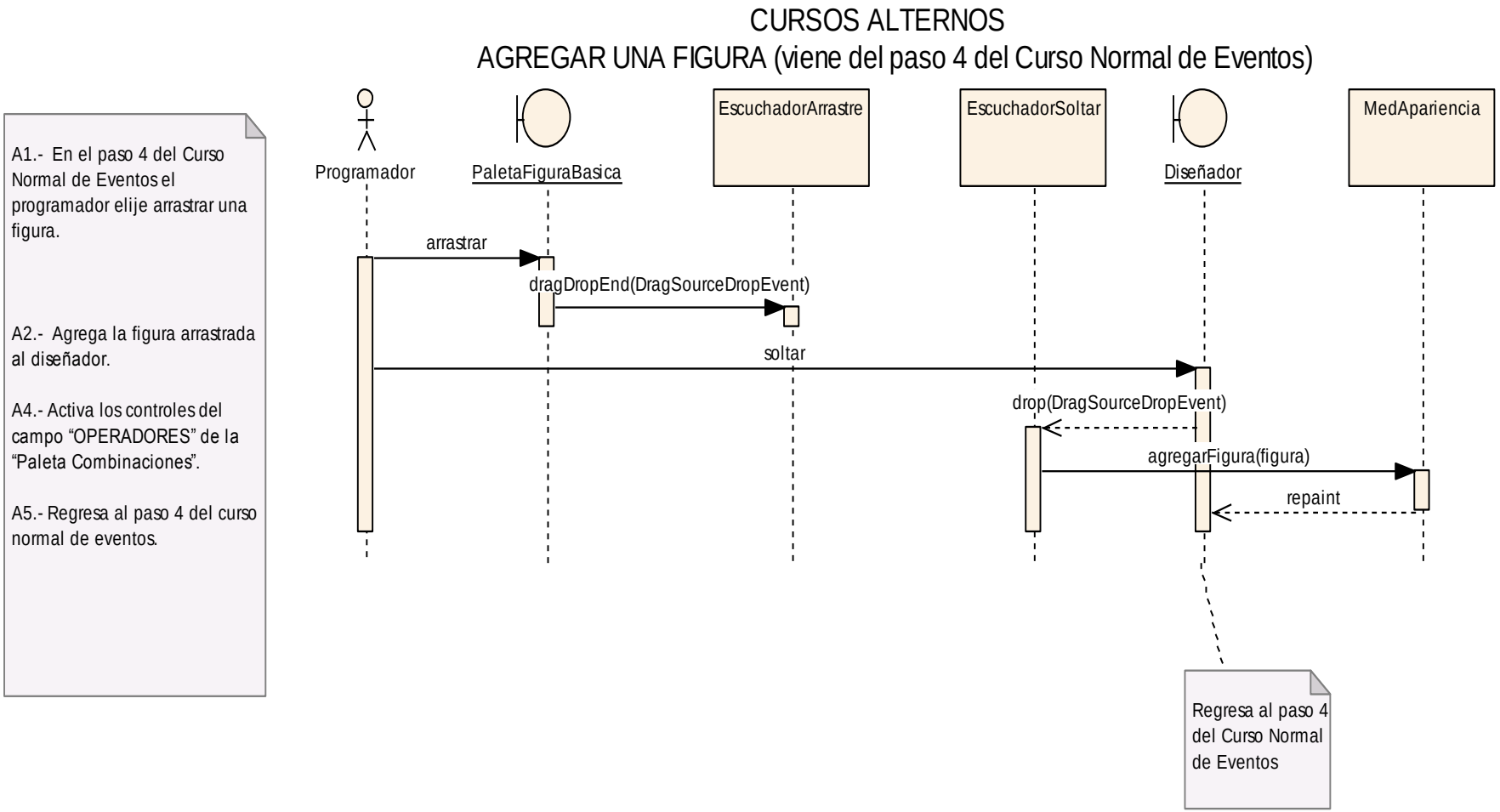



Figura 36.- Diagrama Secuencia Curso Alterno A: Arrastrar Figura

CURSO ALTERNO B: COMBINAR FIGURAS

COMBINA FIGURAS (Viene del paso 4 del Curso Normal de Eventos)

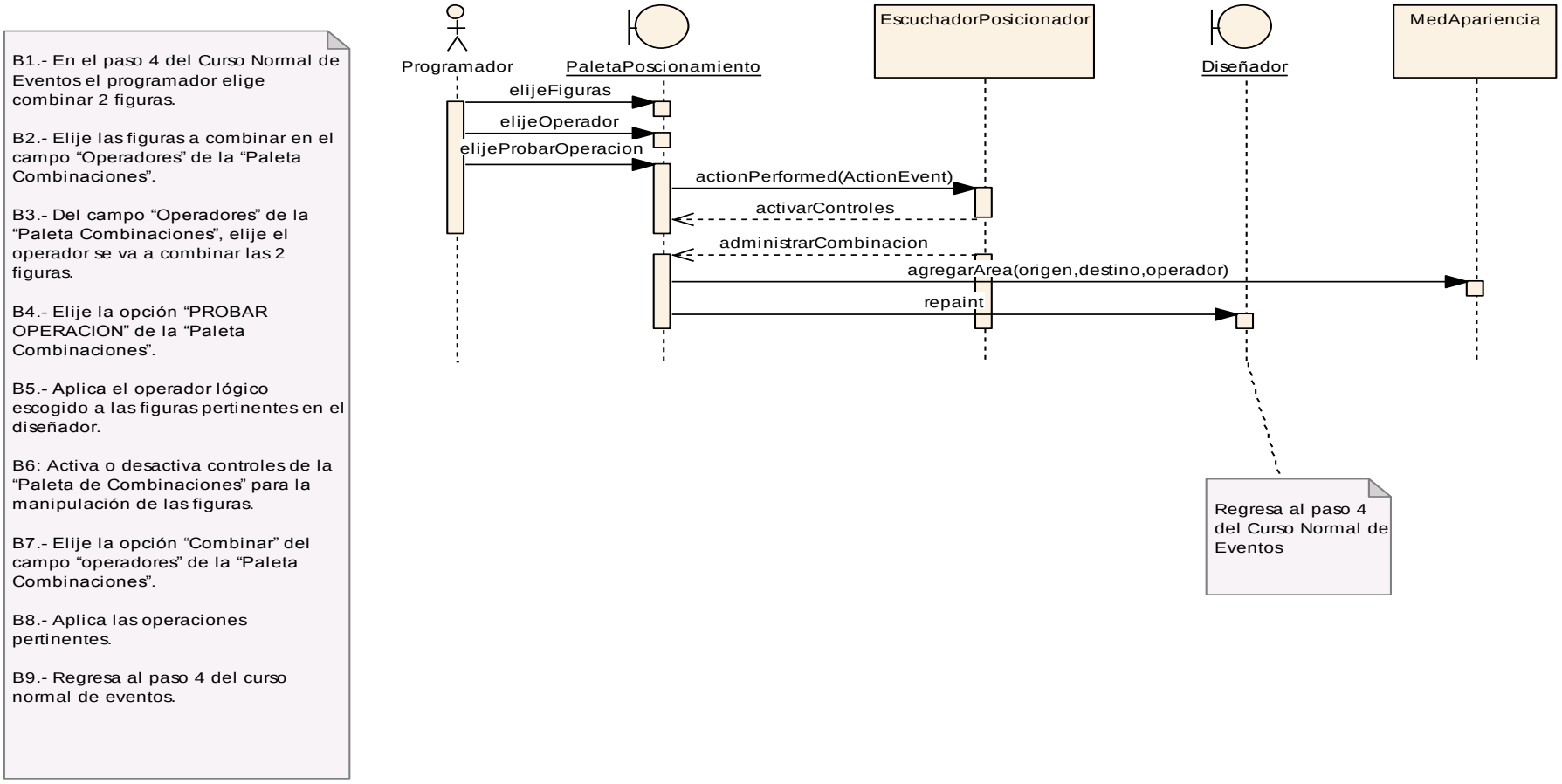



Figura 37.- Diagrama Secuencia Curso Alterno B: Combinar Figuras

CURSO ALTERNO C: MOVER FIGURAS

MOVER UNA FIGURA (Viene del paso 4 del Curso Normal de Eventos)

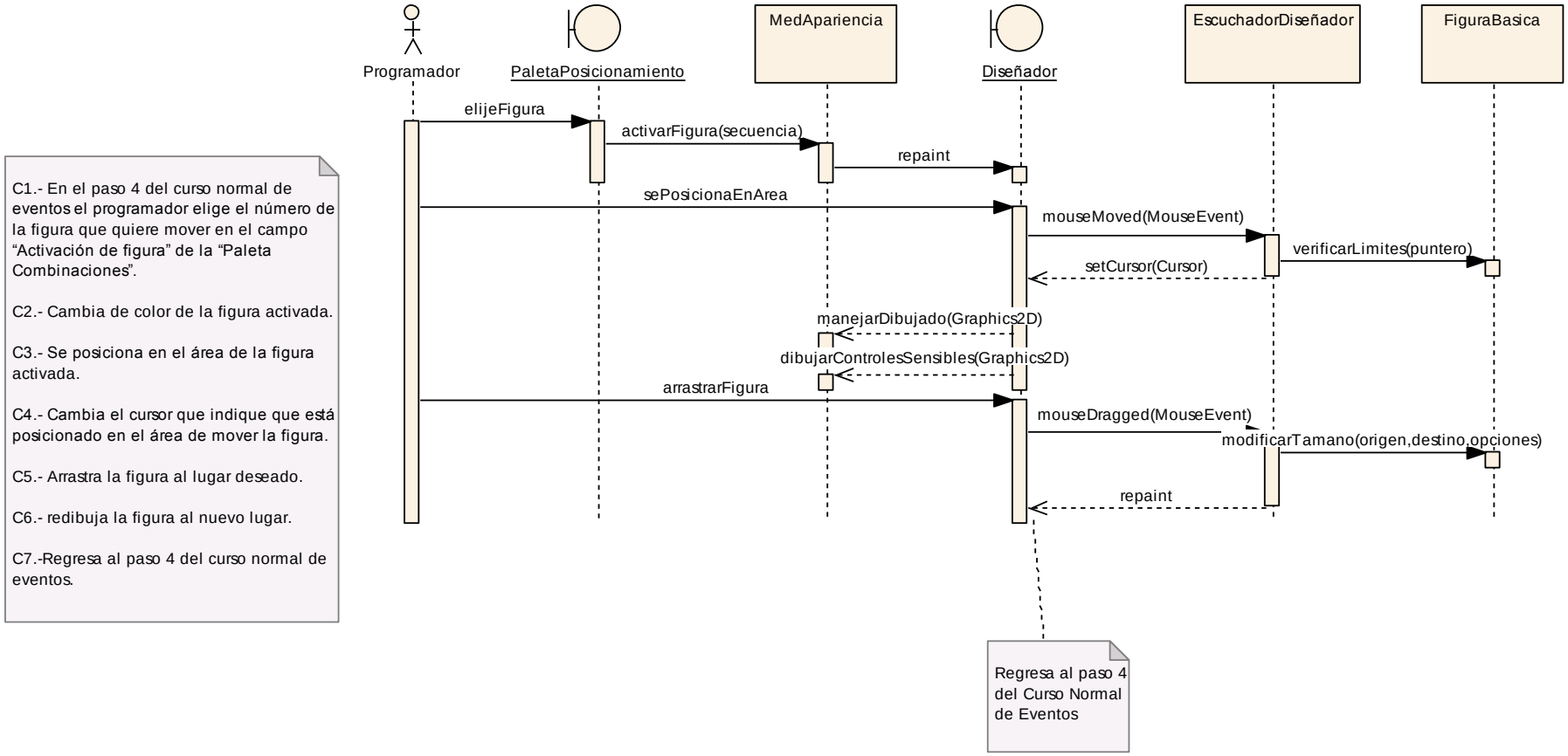



Figura 38.- Diagrama Secuencia Curso Alterno C: Mover Figuras

CURSO ALTERNO D: MODIFICAR TAMAÑO

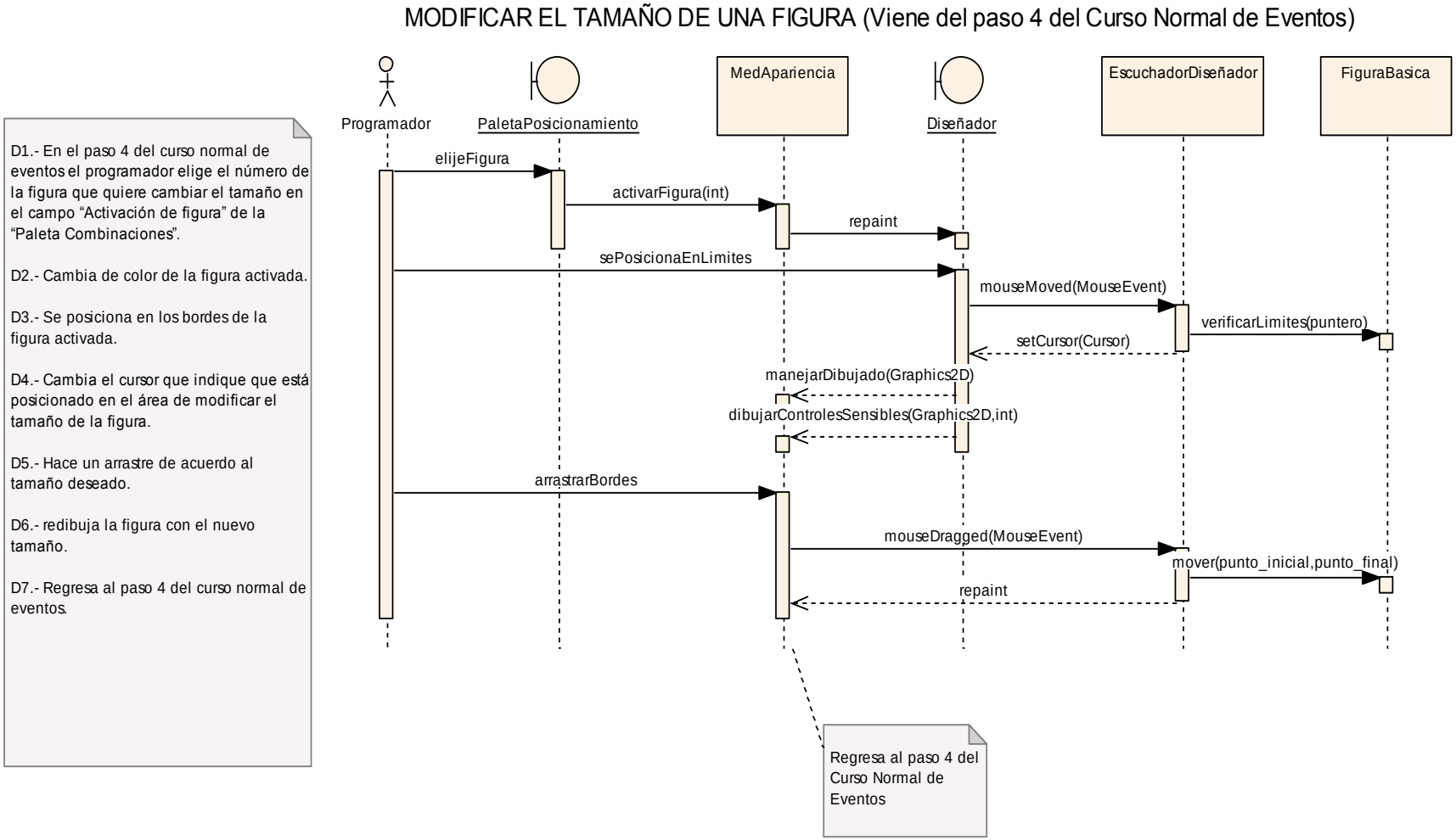



Figura 39.- Diagrama Secuencia Curso Alterno D:Modificar Tamaño.

#### 6.3.2.4. Use Case: Agregar Efectos.

| | |
|---|---|
| **Nombre:** | **Agregar Efectos** |
| **Actor(es):** | Programador |
| **Propósito:** | Proporcionar las funciones básicas para permitir la implementación de efectos para los componentes creados. |
| **Visión General** | Proporcionar efectos que se les puede aplicar a los nuevos componentes creados. |
| **Tipo:** | Opcional, esencial |
| **Referencias:** | RF02. |
| **Precondición(es)** | Tener un componente a personalizar dentro del Diseñador. |
| **Post-condiciones(es)** | |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Programador** | **Sistema** |
| 1.- El programador activa la "Paleta de Efectos". | 2.- Presenta los efectos que se pueden aplicar a los componentes. |
| 3.- Elige el efecto degradado (horizontal, vertical, diagonal izquierda, diagonal derecha) de la "Paleta de Efectos". | 4.- Presenta el Diálogo de Colores. |
| 5.- Elige el número de colores del degradado. | |
| 6.- Elige la opción "Aplicar" del Dialogo de Colores. | |
| | 7.- Redibuja el componente de acuerdo al degradado elegido. |
| | 8.- Ejecuta el use case "Generar Código" |

| | |
|---|---|
| | 9.- Informa sobre la generación de código.<br><br>10.- El use case finaliza. |
| CURSO ALTERNO DE EVENTOS | |
| **A.- ELIGE DEGRADADO 3D.** | |
| A1.- En el paso 3 del curso normal de eventos, el programador elige el degradado 3D en la Paleta de Efectos OPCIÓN GRADIENTES.<br><br>A3.- Elige los colores en el dialogo de Colores.<br><br>A4.- Regresa al paso 6 del curso normal de eventos. | A2.- Presenta el Diálogo de Colores. |
| **B.- ELIGE TRANSPARENCIA** | |
| B1.- En el paso 3 del curso normal de eventos el programador activa el efecto transparencia de la Paleta de Efectos OPCIÓN TRANSPARENCIAS.<br><br>B2.- Ingresa el valor de la transparencia en la Paleta de Efectos OPCION TRANSPARENCIAS.<br><br>B3. Elige la opción “Aplicar” de la “Paleta Efectos” OPCION TRANSPARENCIAS. | B4.- Regresa al paso 7 del curso normal de |

| | |
|---|---|
| | eventos. |
| **C.- ELIGE ACHATAMIENTO** | |
| C1.- En el paso 3 del curso normal de eventos el programador activa el efecto achatamiento de la “Paleta Efectos” OPCIÓN ACHATAMIENTO.<br><br>C2.- Ingresa los valores de los achatamientos en x e y, OPCIÓN ACHATAMIENTO.<br><br>C3.- Elige la opción “Aplicar” en la “Paleta Efectos” OPCIÓN ACHATAMIENTO. | C4.- Redibuja el componente.<br><br>C5.- Regresa al paso 7 del curso normal de eventos. |

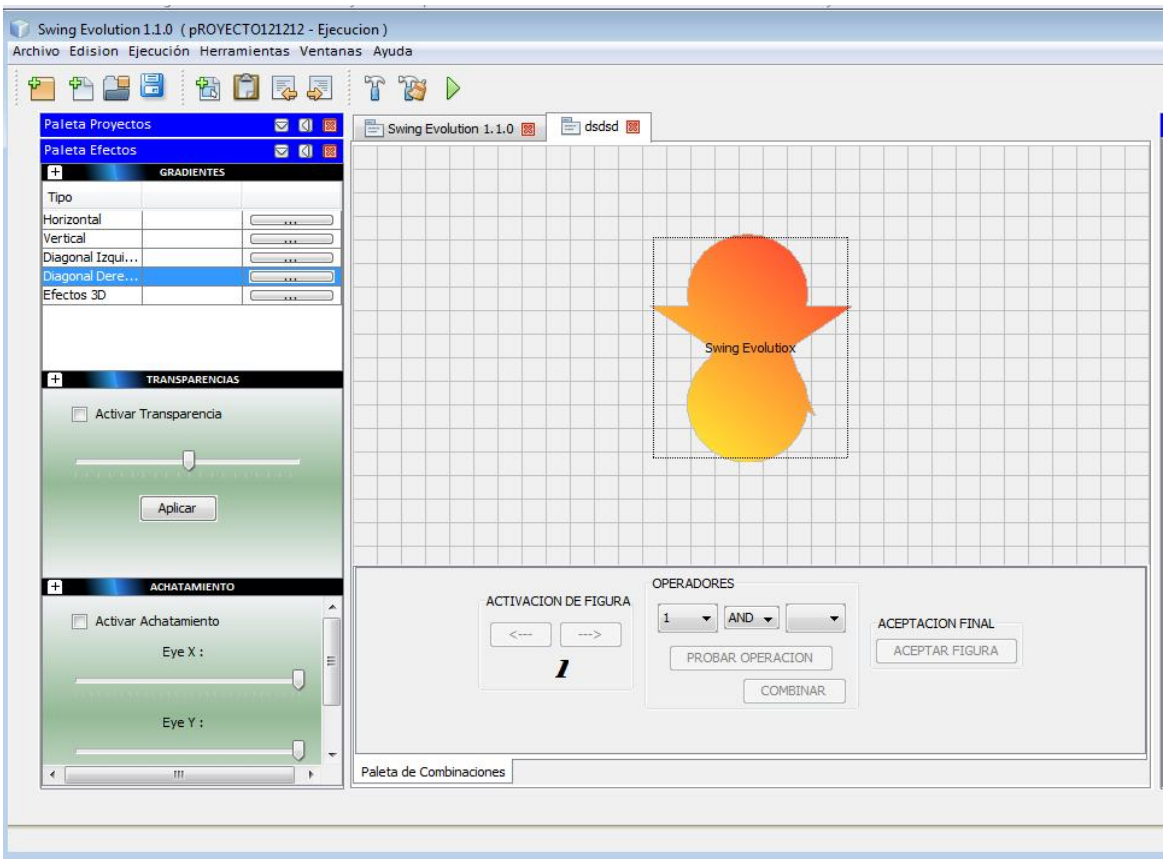



Figura 40. Use Case Agregar Efecto.

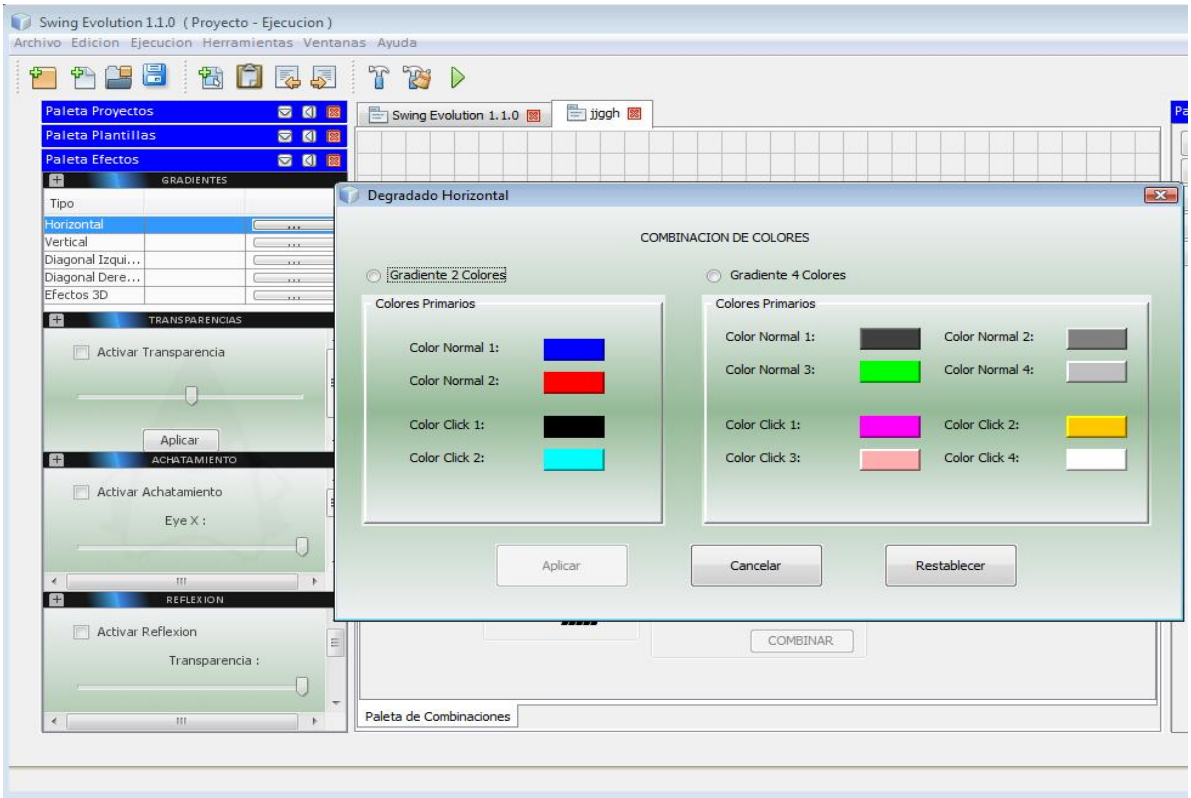


Figura 40 (A). Use Case  Agregar Efecto, Diálogo Colores

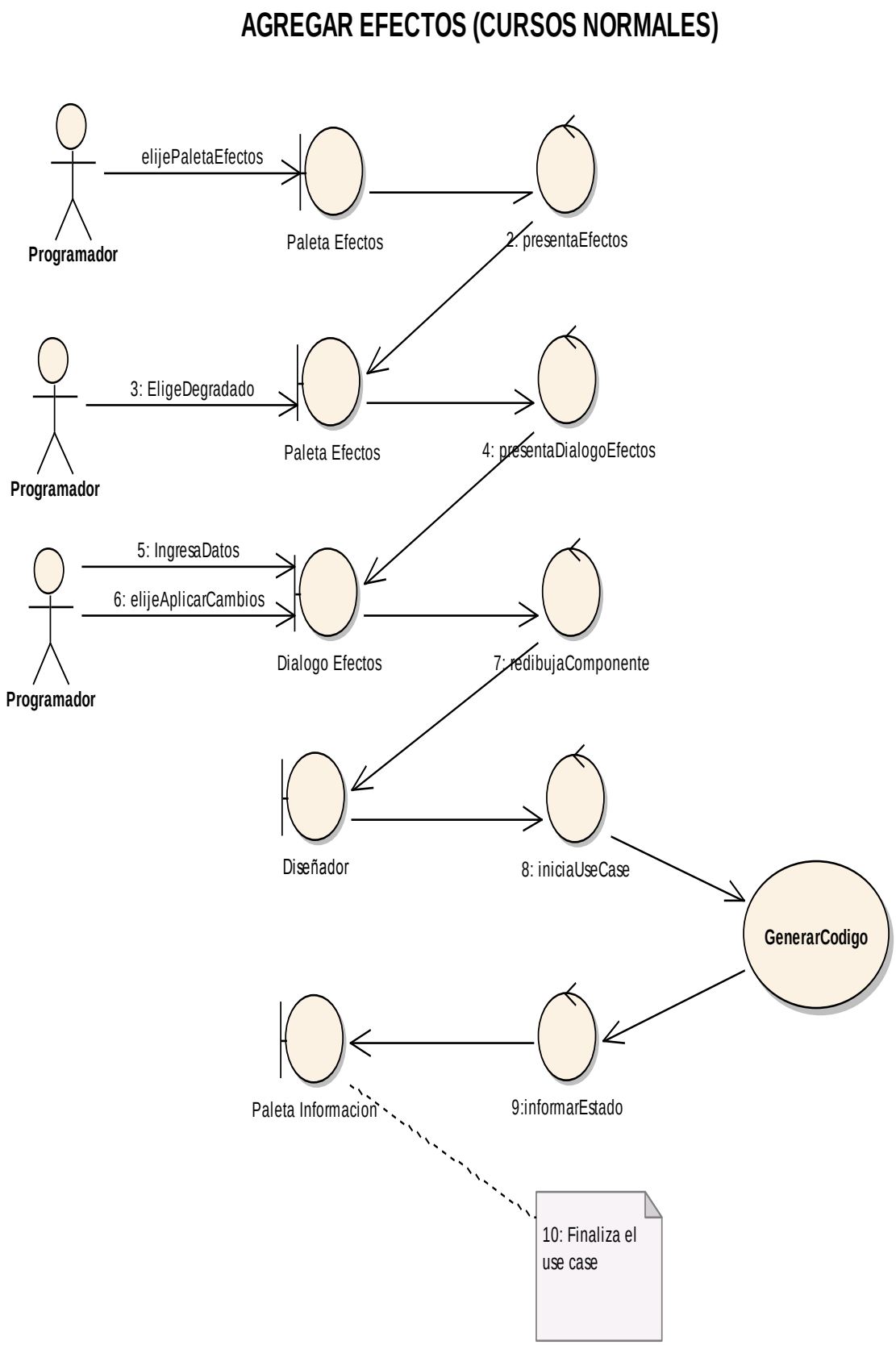


Figura 41.- Curso Normal Agregar Efectos.

## AGREGAR EFECTOS (CURSOS ALTERNOS)

## CURSO ALTERNO A: ELIGE DEGRADADO 3D

A: ELIGE DEGRADADO 3D (Viene del paso 3 del Curso Normal de Eventos)

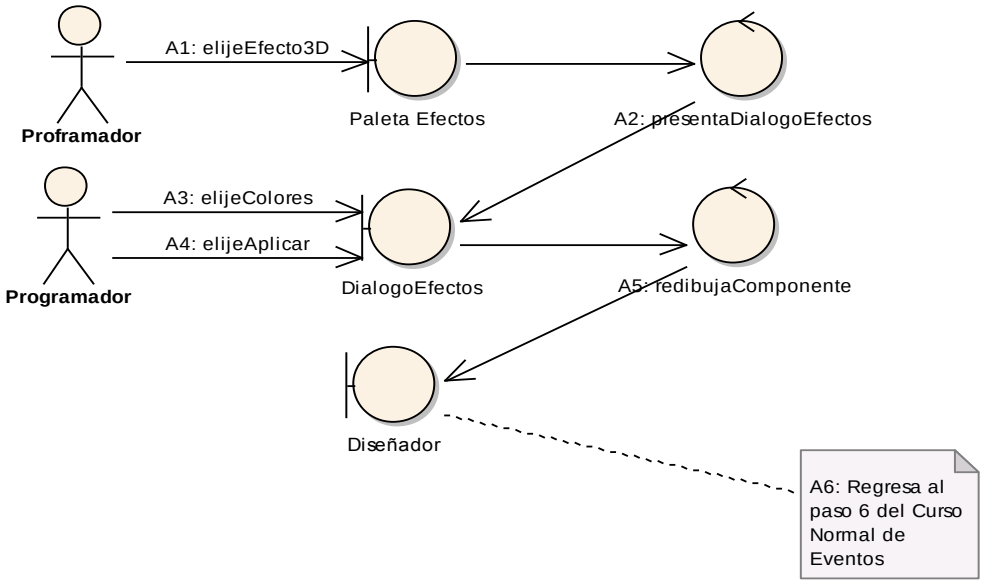



Figura 42.- Curso Alterno A: Elegir degradado 3D.

# CURSO ALTERNO B: ELIGE TRANSPARENCIA

## B: ELIGE TRANSPARENCIA (Viene del paso 3 del Curso Normal de Eventos)

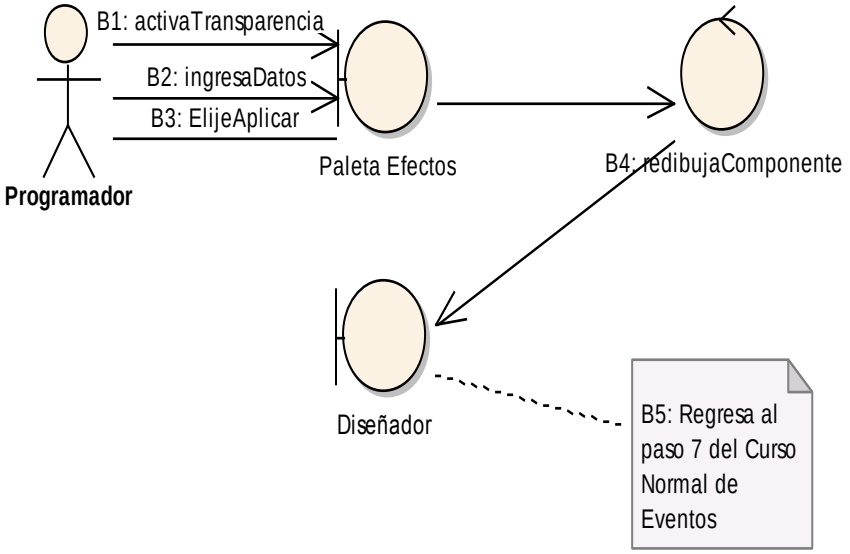



Figura 43.- Curso Alterno B: Elige Transparencia.

# CURSO ALTERNO C: ELIGE ACHATAMIENTO

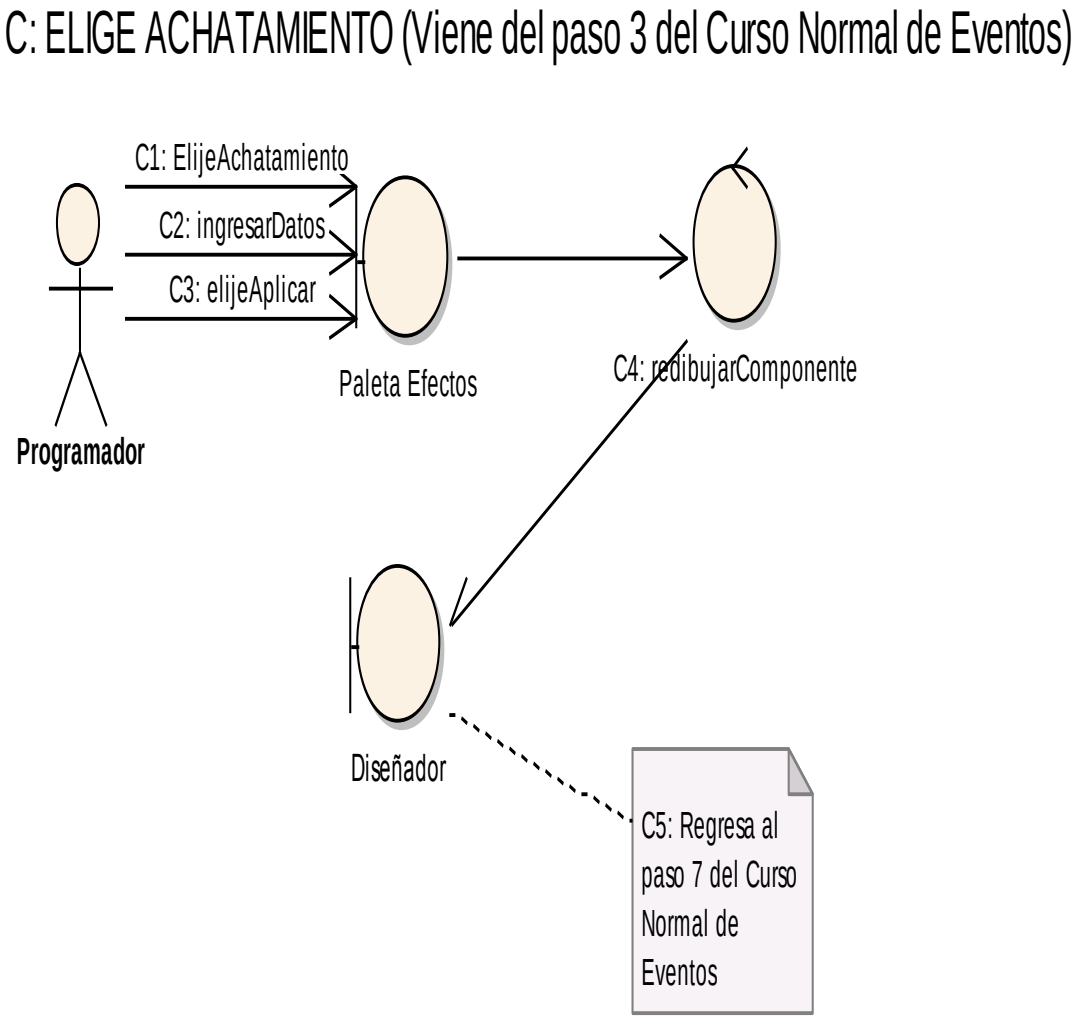



Figura 44.- Curso Alterno C: Elige Achatamiento.

## DIAGRAMA DE SECUENCIA: AGREGAR EFECTOS

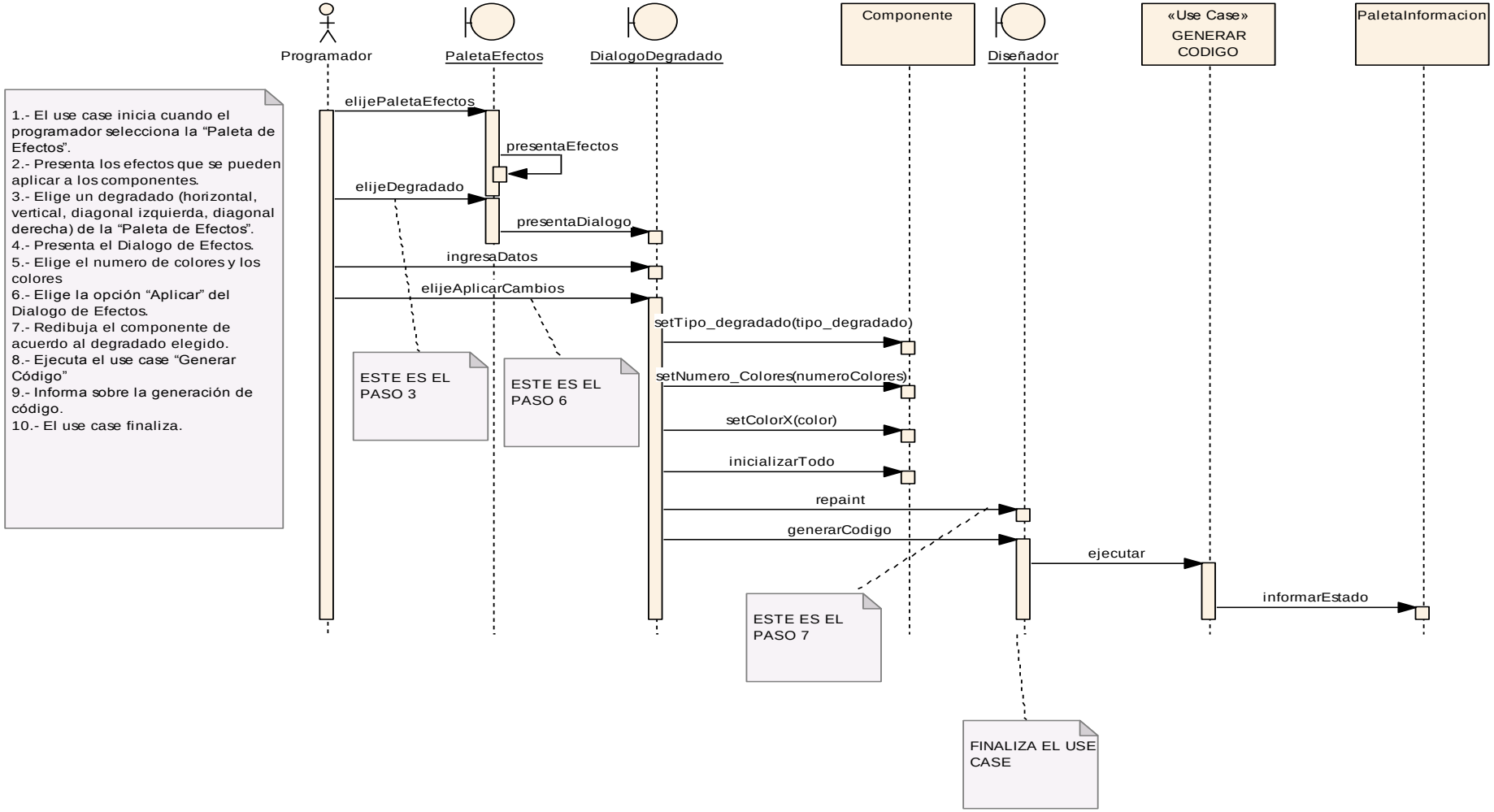



Figura 45.- Diagrama Secuencia Agregar Efectos.

CURSO ALTERNO A: DEGRADADO 3D

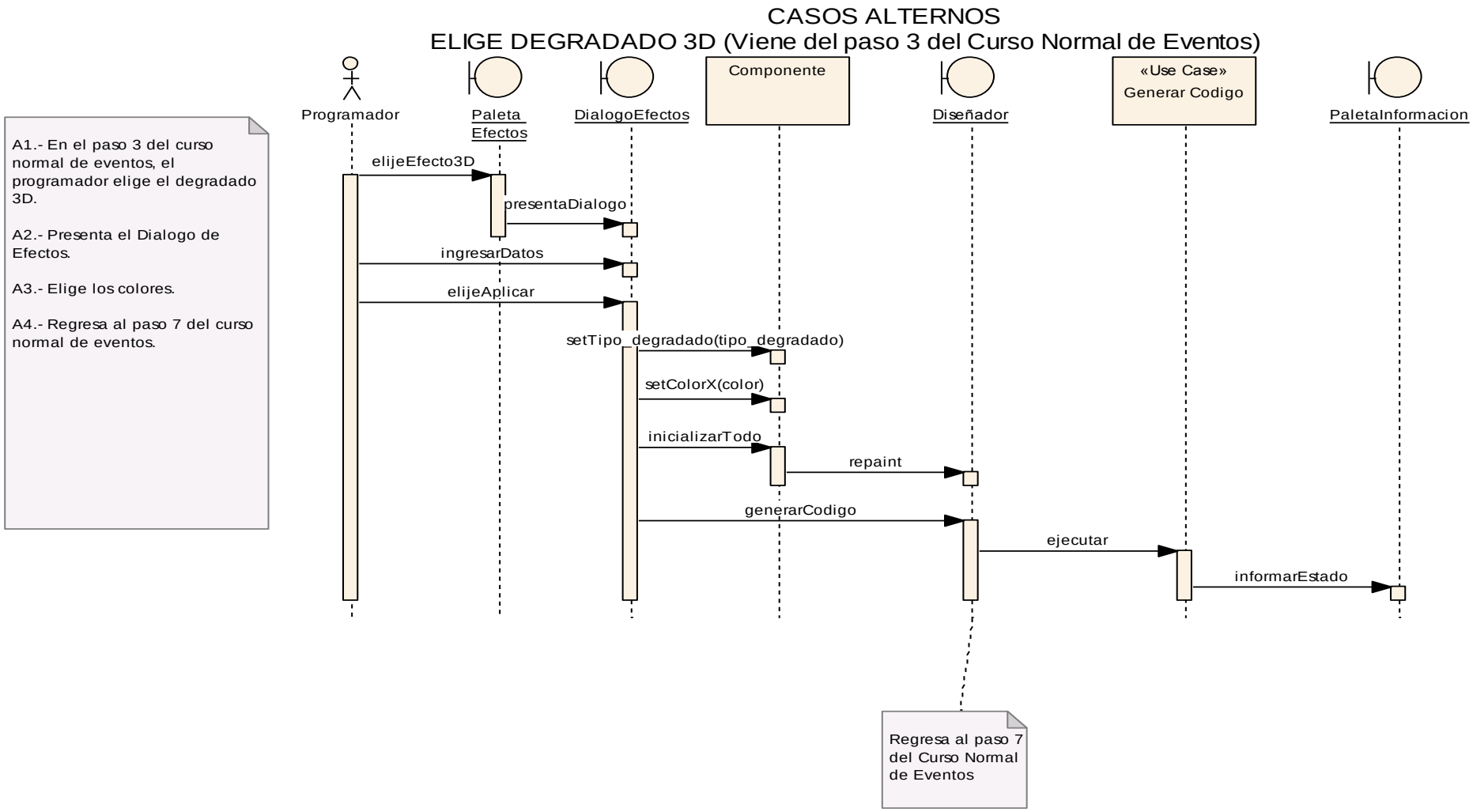


Figura 46.- Diagrama Secuencia Curso Alterno A: Elige Degradado 3D

CURSO ALTERNO B: ELIGE TRANSPARENCIA

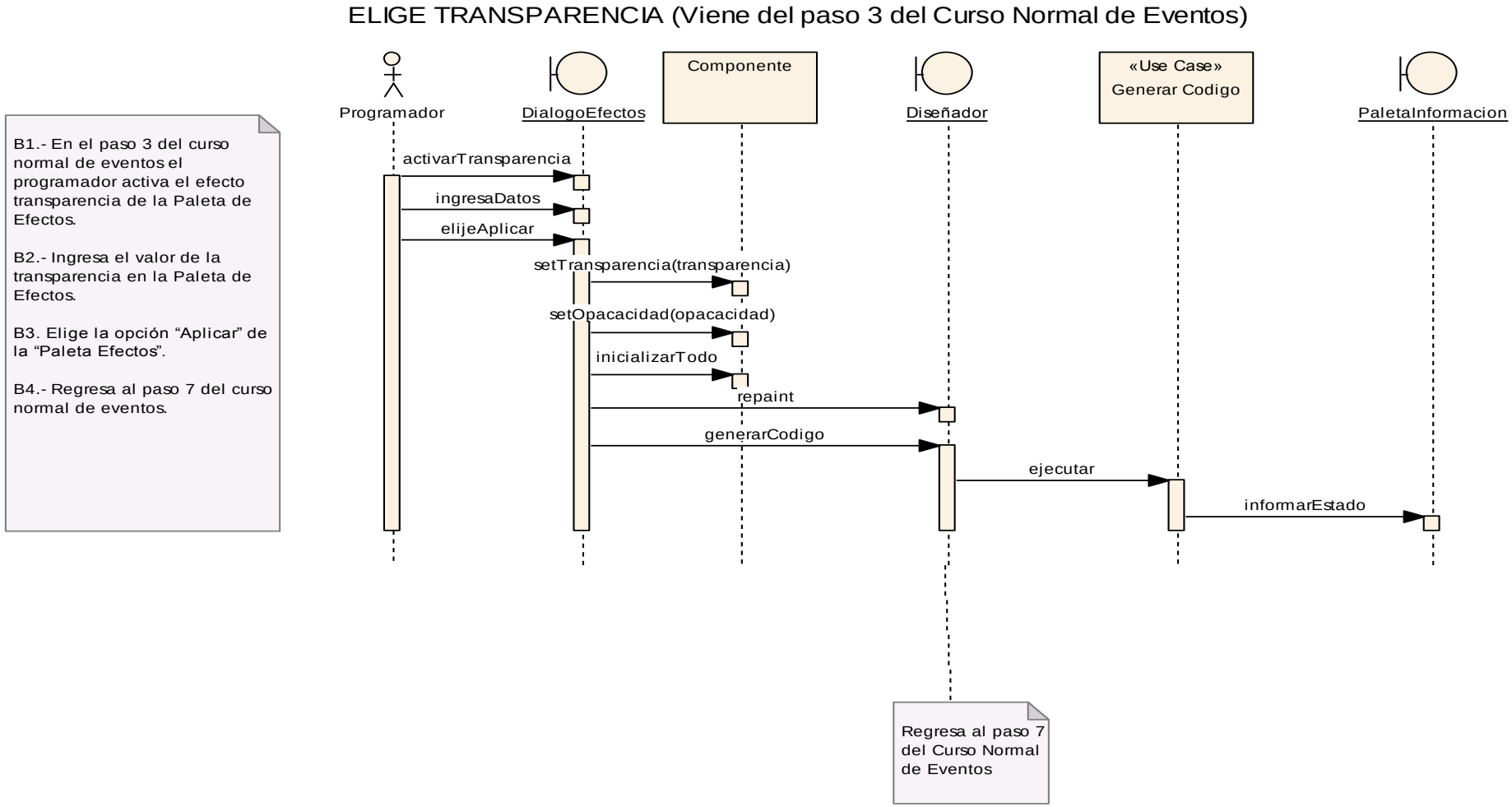



Figura 47.- Diagrama Secuencia Curso Alterno B: Elige Transparencia.

CURSO ALTERNO C: ELIGE ACHATAMIENTO

ELIGE ACHATAMIENTO (Viene del paso 3 del Curso Normal de Eventos)

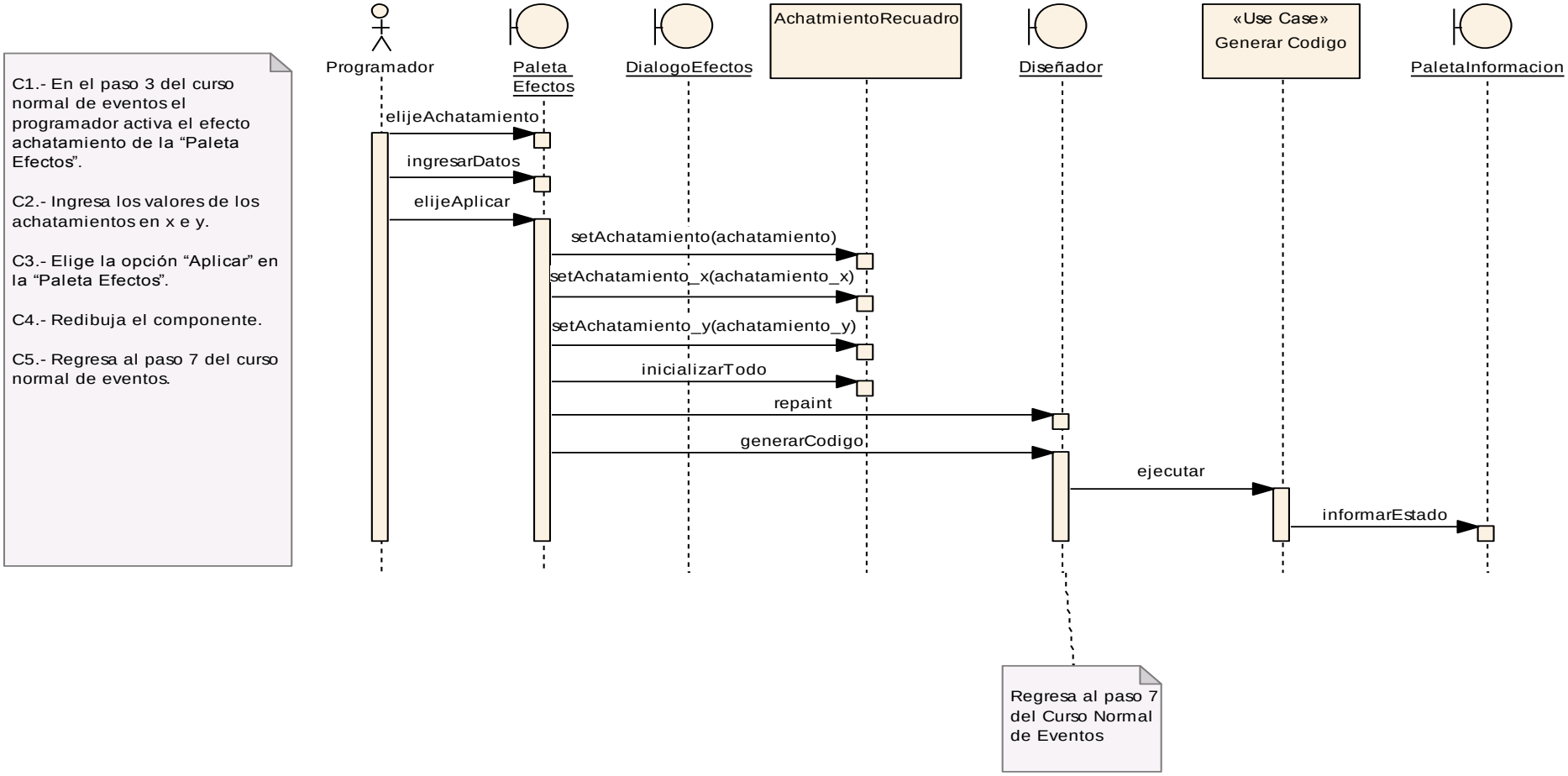


Figura 48.- Diagrama Secuencia Curso Alterno C: Elige Achatamiento.

#### 6.3.2.5. Use Case: Generar Módulo.

| | |
|---|---|
| **Nombre:** | **Generar Módulo** |
| **Actor(es):** | Programador |
| **Propósito:** | Proporcionar las funciones básicas para la creación de un módulo (.jar) |
| **Visión General** | Después de haber creado el nuevo componente se creará un módulo para ser implementado en la plataforma NetBeans 5.0 y posteriores. |
| **Tipo:** | Primario, esencial |
| **Referencias:** | RF12, RF16. |
| **Precondición(es)** | Debe existir un proyecto abierto con un componente creado como mínimo. |
| **Post-condiciones(es)** | Instalación del módulo .jar |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Programador** | **Sistema** |
| 1.- El use empieza cuando el programador elije la opción "Clean & Build" en el Popup Menu de la Paleta Proyectos. | 2.- Busca el archivo Build.xml en la ruta del proyecto en Ejecución.<br><br>3.- Visualiza la Paleta de Información, en donde se muestra los LOGS de salida de la generación del Modulo .Jar.<br><br>4.- El sistema ejecuta el use case "Generar Archivos de configuración".<br><br>5.- Informa sobre la generación de archivos de configuración a la Paleta de Ejecución.<br><br>6.- Realiza la eliminación de los directorios "dist","lib","resource", los vuelve a crear y compila las fuentes .JAVA.<br><br>7.- Informa sobre la compilación de los archivos .java a la Paleta de Información.<br><br>8. Crear el ejecutable .jar incluyendo los archivos de configuración y las clases |

| | |
|---|---|
| | compiladas.<br><br>9.- Informa la creación del ejecutable .jar a la Paleta de Información.<br><br>10.- Finaliza el use case. |
| **CURSO ALTERNO DE EVENTOS** | |
| **A.- RECOMPILAR JAVABEAN INSTALADO** | |
| A1.-En el paso 1 del Curso Normal de Eventos el Programador decide una vez instalado el archivo .JAR del Proyecto en la Plataforma Netbeans volver a "Clean & Build" el Proyecto utilizado. | A2.- El Sistema deberá esperar que la plataforma Netbeans deje de utilizar el .JAR del Proyecto añadido a la plataforma para su correcta generación del Modulo.<br><br>A3.- Regresa al Paso 2 del Curso Normal de Eventos. |

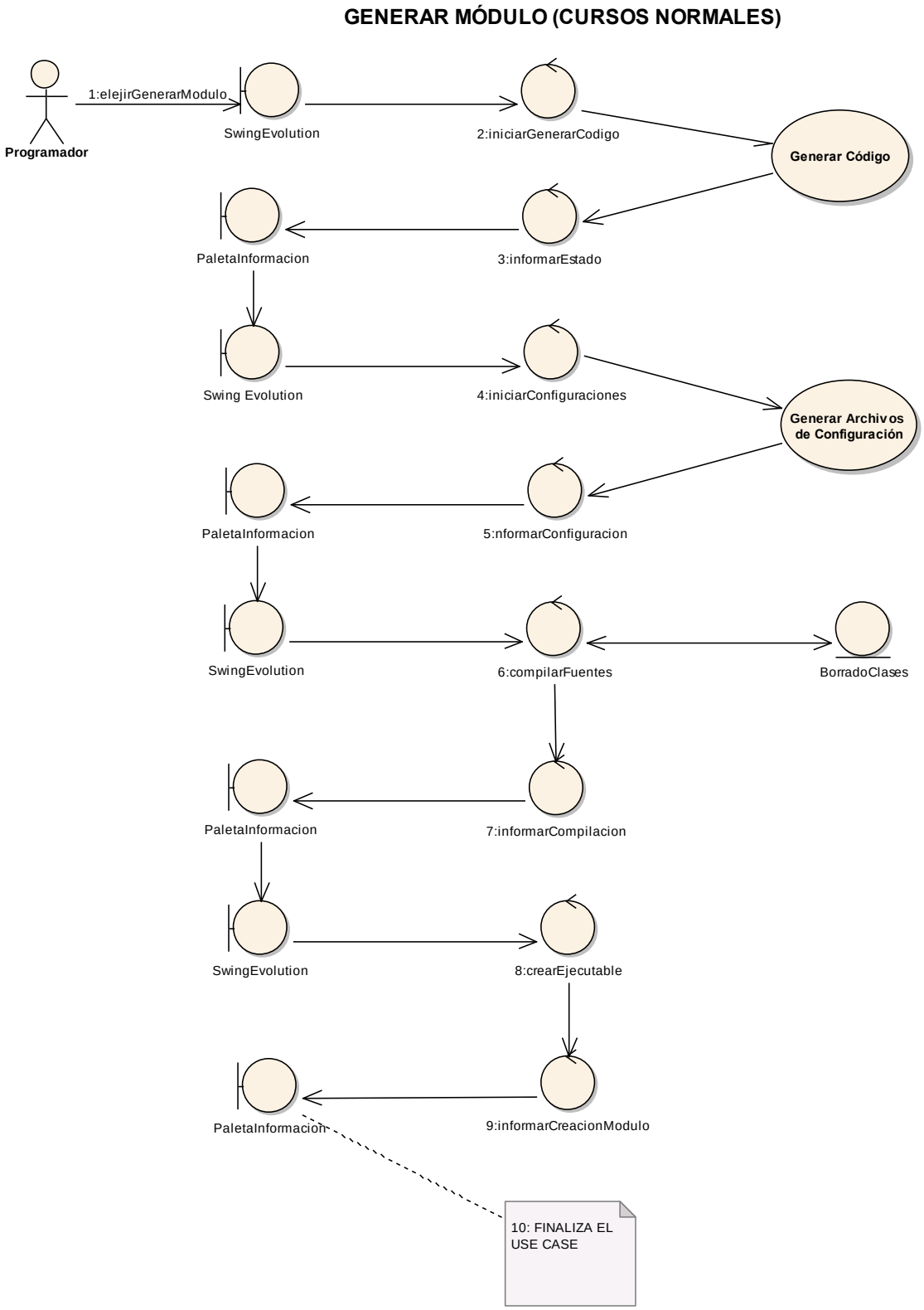



Figura 49.- Curso Normal Generar Módulo.

## CURSO ALTERNO A: RECOMPILAR JAVABEAN INSTALADO

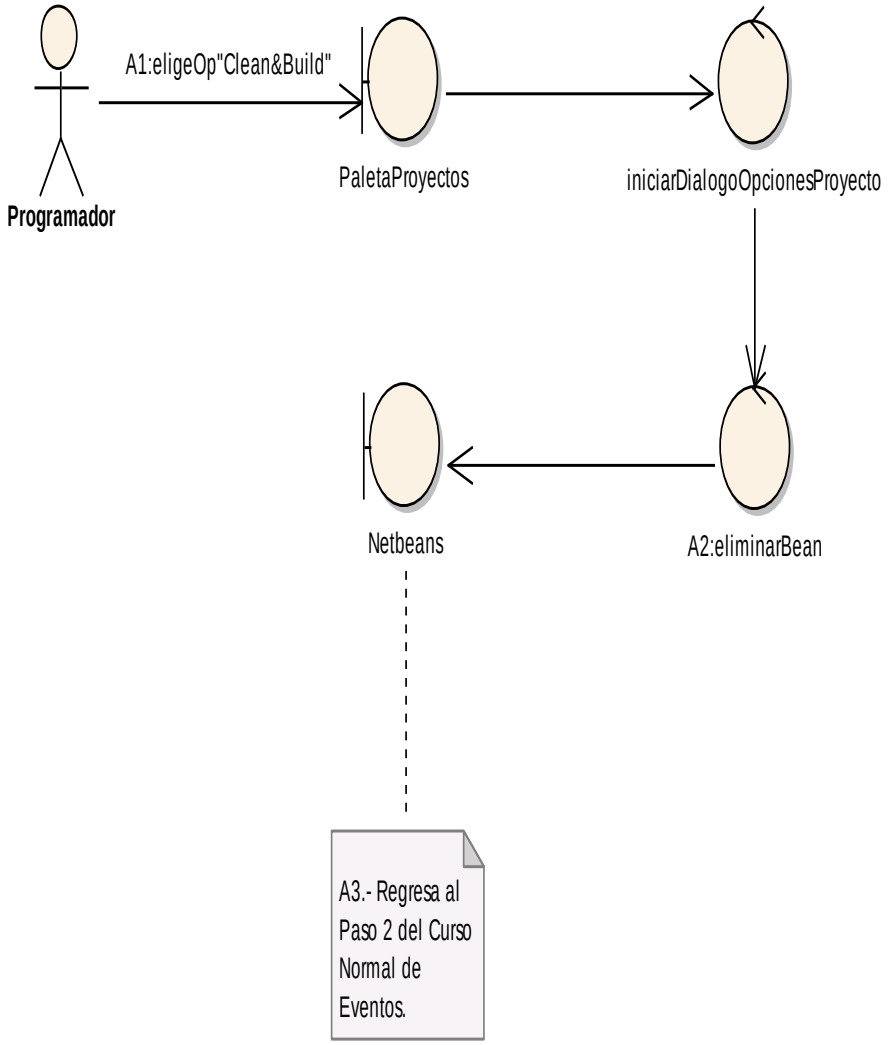



Figura 50.- Curso Alterno A: Recompilar JavaBean Instalado.

DIAGRAMA DE SECUENCIA (GENERAR MODULO)

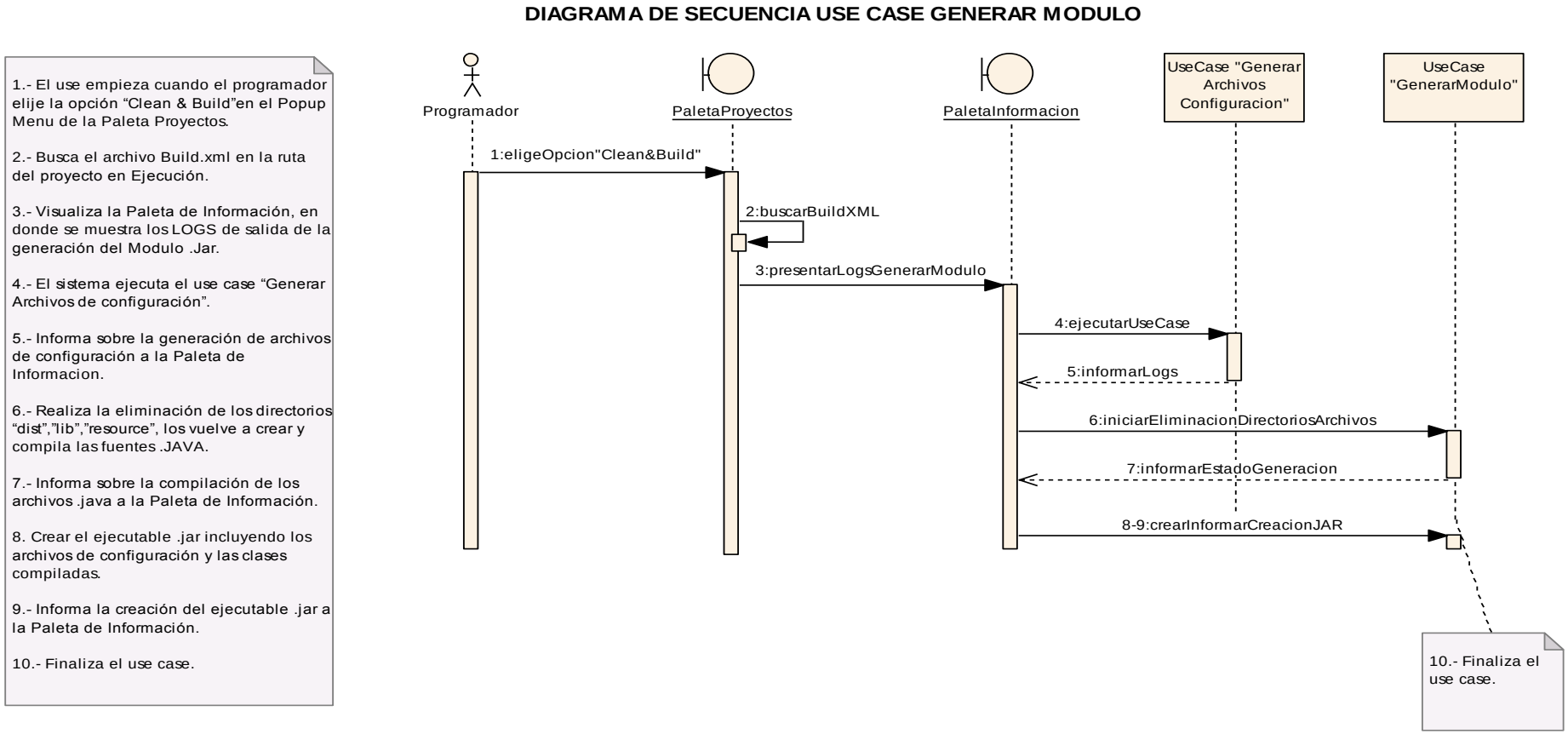



Figura 51.- Diagrama Secuencia de Generar Modulo.

CURSO ALTERNO A: (RECOMPILAR PROYECTO)

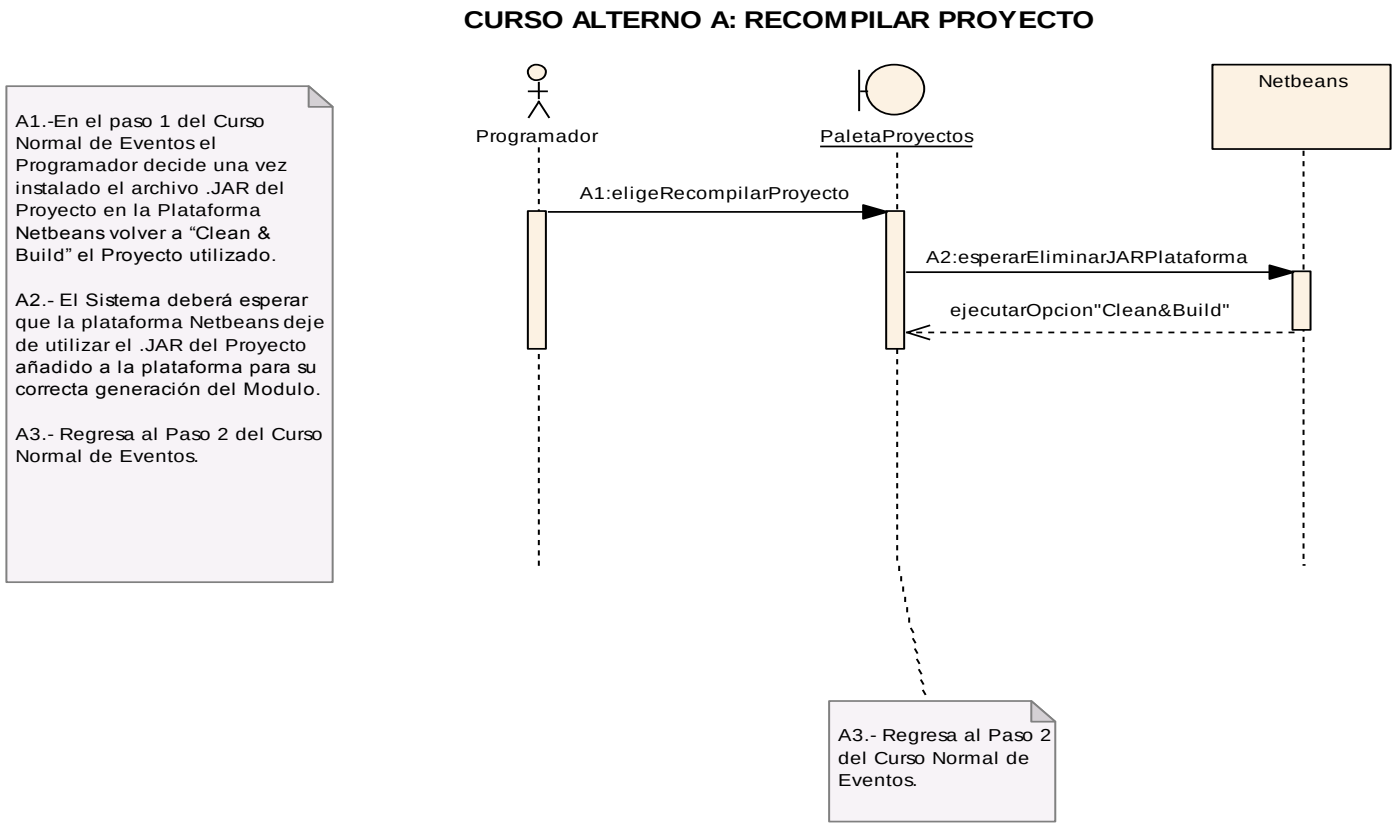



Figura 52.- Diagrama Secuencia Curso Alterno A: Recompilar Proyecto.

### 6.3.2.6. Use Case: Administrar Directorios y Archivos.

| | |
|---|---|
| **Nombre:** | **Administrar Directorios y Archivos** |
| **Actor(es):** | Administrar Proyecto o Administrar Plantilla |
| **Propósito:** | Crear y borrar la estructura de un proyecto o plantilla organizado por directorios y archivos. |
| **Visión General** | Es el encargado de trabajar con los directorios y archivos de un proyecto o plantilla como es la creación y el borrado de los mismos. |
| **Tipo:** | Primario, Esencial |
| **Referencias:** | RF13, RF14, RF15. |
| **Precondición(es)** | Haber iniciado el use case ”Administrar Proyecto” o “Administrar Plantilla” |
| **Post-condiciones(es)** | |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Actor** | **Sistema** |
| 1.- Este Use Case inicia cuando el programador inicia el “Use Case Administrar Proyecto” y este se ejecuta. | 2.- Genera el árbol de directorios necesarios para el Proyecto, según las especificaciones de la plataforma Netbeans.<br><br>3.- Se inicia el Use Case Generar Archivos de Configuración.<br><br>4.- Informar estado de la creación de directorios y archivos a la Paleta de Información.<br><br>5.- Finaliza el use case. |

| **CURSO ALTERNO DE EVENTOS** | |
|---|---|
| **A.- INICIAR_UC_ADMINISTRAR_PLANTILLA** | |
| A1.- En el paso 1 del Curso Normal de Eventos el programador inicia el Use Case | A2.- Genera el árbol de directorios necesarios para la Plantilla, según las especificaciones |

| | |
|---|---|
| “Administrar Plantilla” y este se ejecuta. | propias de la plataforma Swing Evolution.<br><br>A3.- Se inicia el Use Case Generar Archivos de Configuración.<br><br>A4.- Informa sobre la generación y creación de directorios y archivos a la Paleta de Información.<br><br>A5.- Regresa al paso 5 del Curso Normal de Eventos. |

**ADMINISTRAR DIRECTORIOS Y ARCHIVOS (CURSOS NORMALES)**

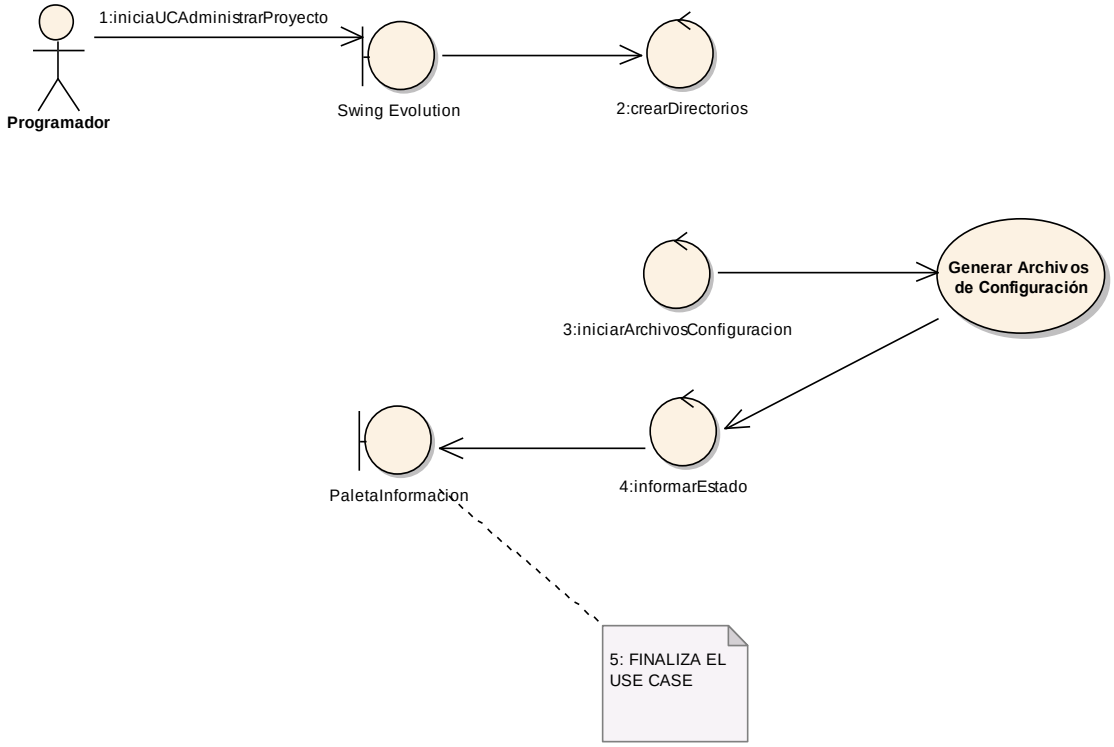



Figura 53.- Curso Normal: Administrar Directorios y Archivos

**Administrar directorios y Archivos (cursos alternos)**

**CURSO ALTERNO : A INICIAR_UC_ADMINISTRAR_PLANTILLA**

Viene del paso 1 del Curso Normal de Eventos

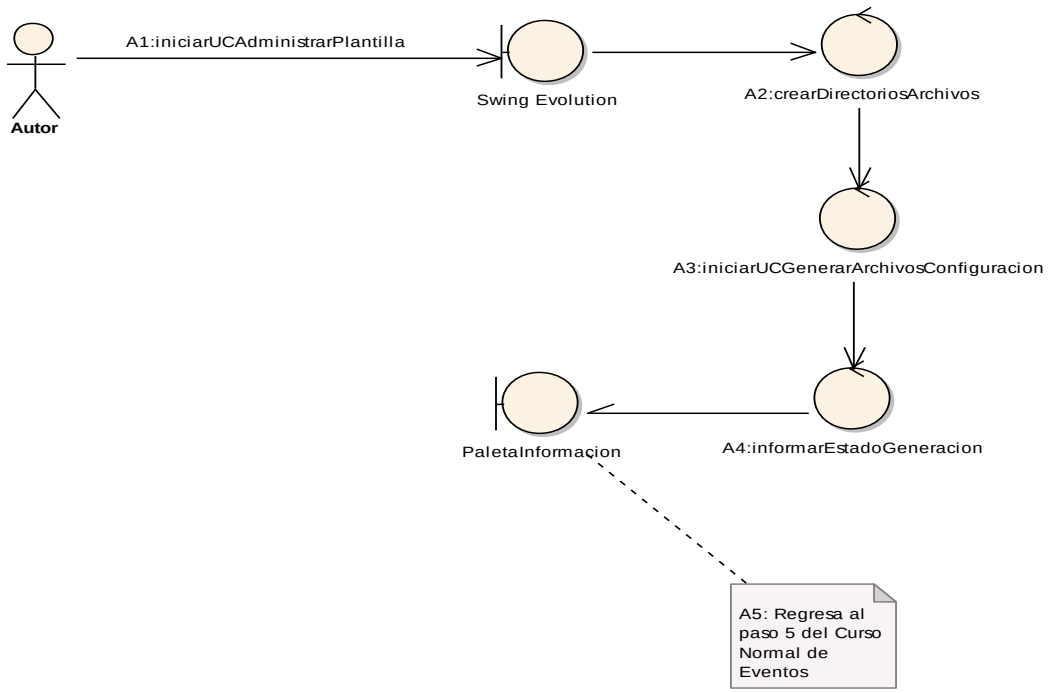



Figura 54.- Curso Alterno A: Iniciar UC Administrar Plantilla.

DIAGRAMA DE SECUENCIA (ADMINISTRAR DIRECTORIOS Y ARCHIVOS)

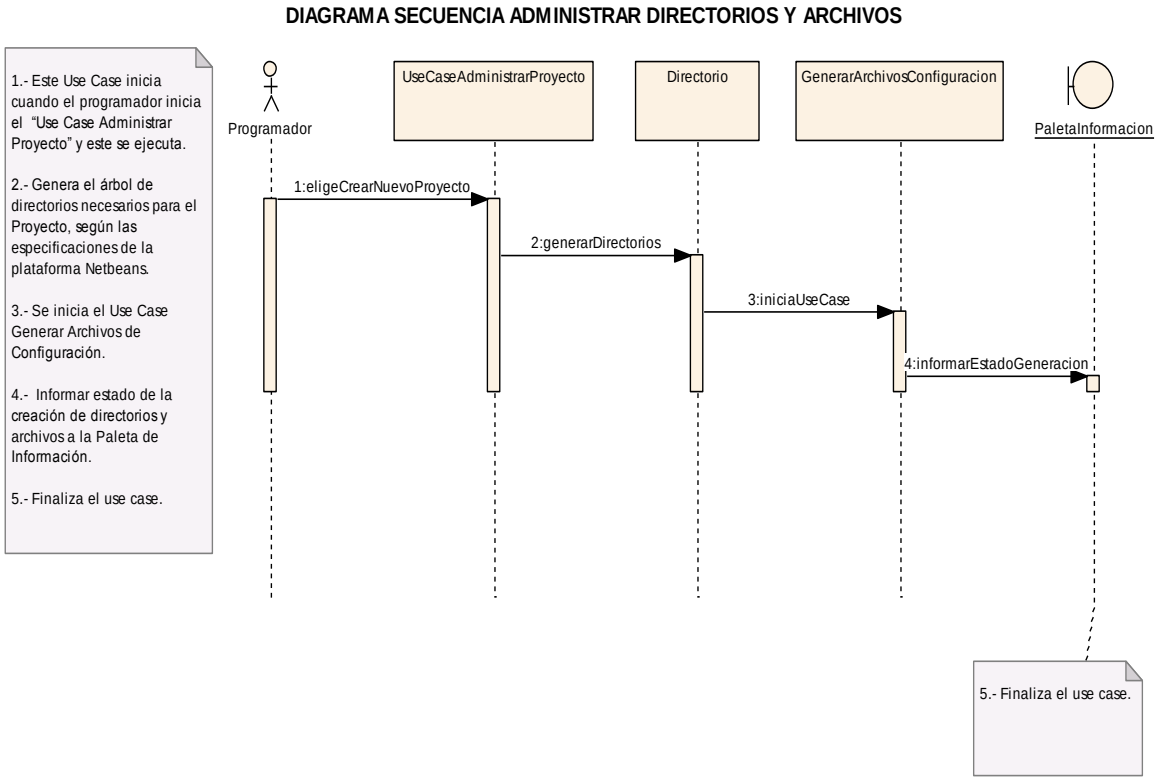



Figura 54 (A).- Diagrama de Secuencia Administrar Directorios y Archivos

CURSO ALTERNO A: INICIAR UC ADMINISTRAR PLANTILLA

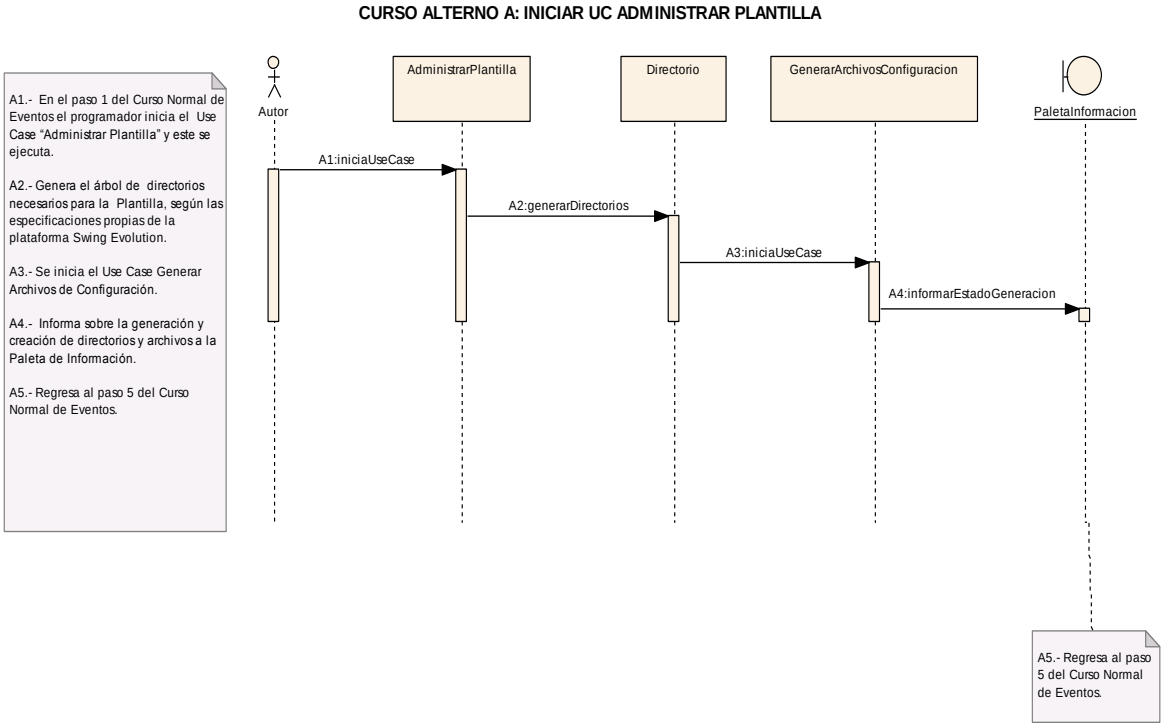



Figura 54(B).- Diagrama Secuencia del Curso Alterno A: Iniciar UC Administrar Plantilla.

### 6.3.2.7. Use Case: Generar Archivos de Configuración.

| | |
|---|---|
| **Nombre:** | **Generar Archivos de Configuración** |
| **Actor(es):** | Programador |
| **Propósito:** | Generar archivos necesarios para la migración de los nuevos componentes a la plataforma Netbeans. |
| **Visión General** | Generar Archivos según las especificaciones de la Plataforma Netbeans. Tareas Ant (Apache). |
| **Tipo:** | Primario, esencial |
| **Referencias:** | RF16. |
| **Precondición(es)** | Haber realizado el Use Case "Generar Módulo" |
| **Post-condiciones(es)** | |
| **CURSO NORMAL DE EVENTOS** | |
| **Programador** | **Sistema** |
| 1.- El use case inicia cuando el programador ejecuta el Use Case "Generar Módulo". | 2.- Crea un archivo de configuración Build.xml.<br>3.- Llama al subsistema Ant-Apache para la realización de las Tareas Ant, que existen dentro del Archivo Build.xml en cada proyecto y lo guarda.<br>4.- Crea un Archivo de Manifiesto.mf, según las especificaciones propias de Ant-Apache y lo guarda.<br>5.- Informa sobre la generación de archivos de configuración.<br>6.- Use case Finaliza. |
| **CURSO ALTERNO DE EVENTOS** | |
| | |

## GENERAR ARCHIVOS DE CONFIGURACION (CURSOS NORMALES)

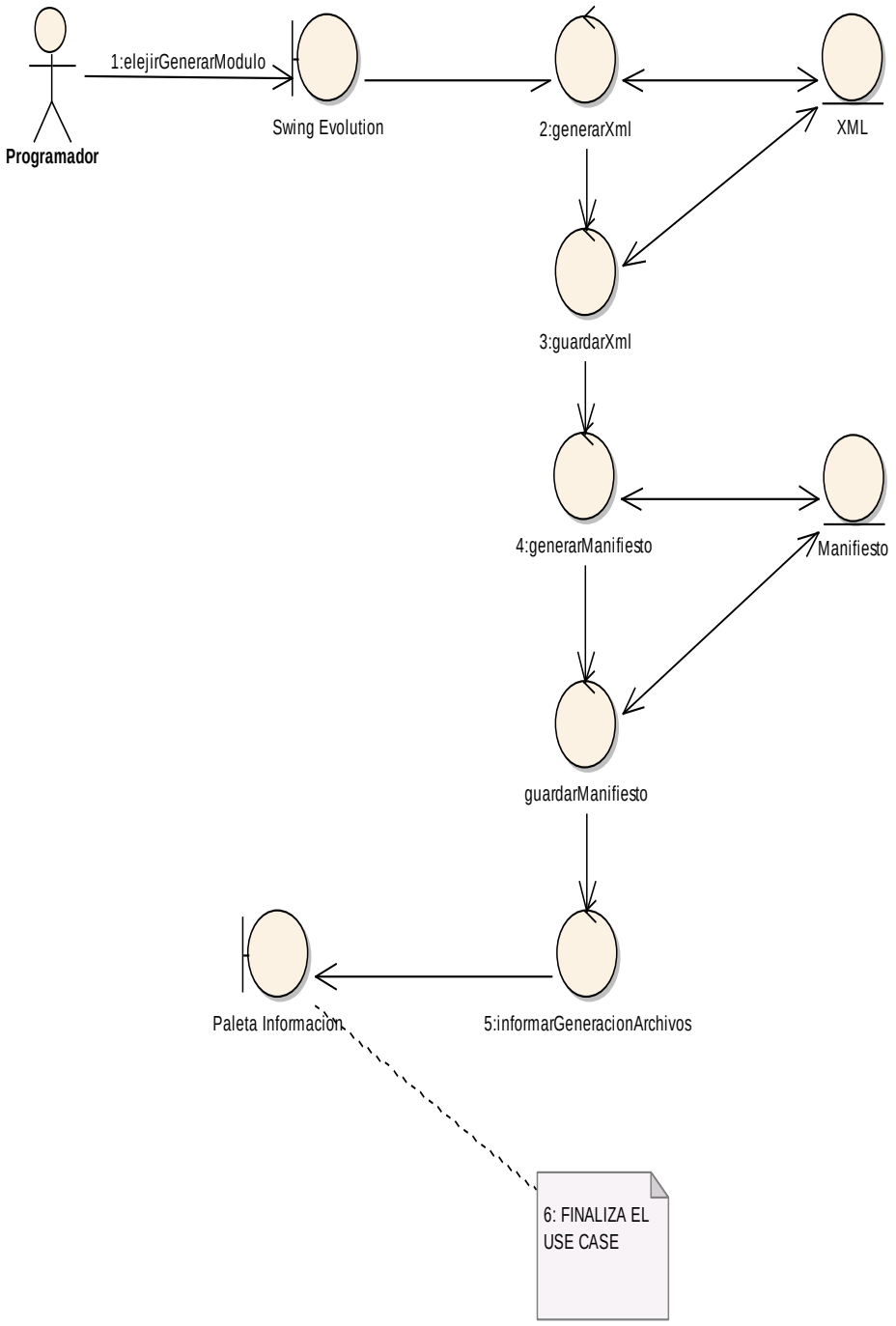



Figura 55.- Curso Normal Generar Archivos Configuración.

DIAGRAMA DE SECUENCIA (GENERAR ARCHIVOS DE CONFIGURACION)

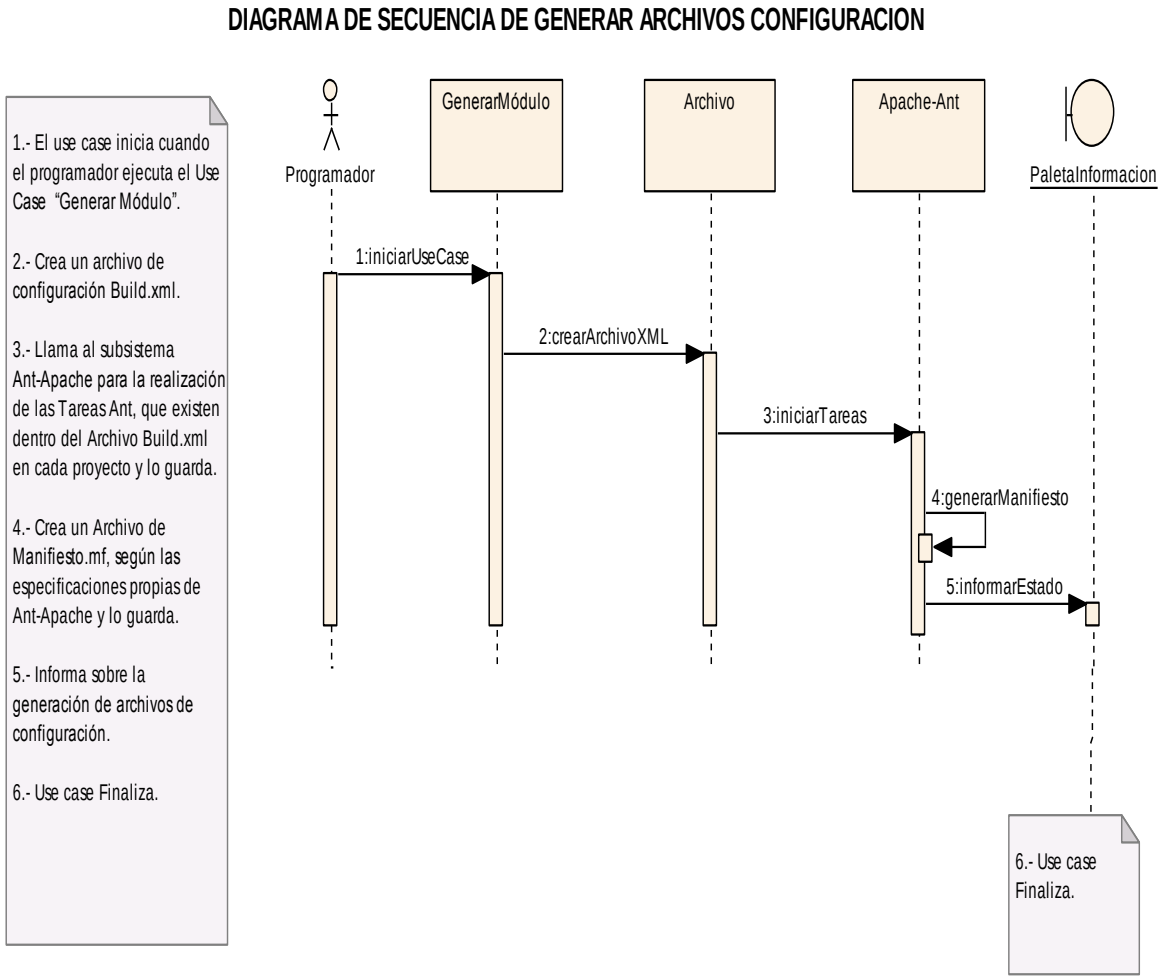



Figura 55(A).- Curso Normal Generar Archivos Configuración.

#### 6.3.2.8. Use Case: Generar Código.

| | |
|---|---|
| **Nombre:** | **Generar Código** |
| **Actor(es):** | Modificar Apariencia, Generar Módulo o Agregar Efectos |
| **Propósito:** | Generar código tanto .XML como .java basado en eventos ocurridos en el diseñador |
| **Visión General** | Generar código de los archivos .java necesarios para construir el o los nuevos componentes Swing. |
| **Tipo:** | Primario, esencial |
| **Referencias:** | RF03. |
| **Precondición(es)** | Sea llamado por algún otro USE CASE |
| **Post-condiciones(es)** | |

| **CURSO NORMAL DE EVENTOS** | |
|---|---|
| **Actor** | **Sistema** |
| 1.- Es llamado por otro Caso de Uso. (Agregar Efecto, Modificar Apariencia, etc.) | |
| | 2.- Recopila toda la información del código que se tiene que generar del componente en el Vector de Instrucciones. |
| | 3.- Se lee la primera posición del Vector de Instrucciones e inicializa los datos principales del componente (nombre del componente, superclase, nombre del proyecto, etc.). |
| | 4. Crea el archivo .java del componente. |
| | 5.- Se lee la siguiente instrucción a realizar del Vector de Instrucciones. |

| | |
|---|---|
| | 6. Finaliza el use case. |
| **CURSO ALTERNO DE EVENTOS** | |
| **A.- INSTRUCCIÓN ENCONTRADA (MODIFICAR_CADENA)** | |
| | A1.- En el paso 6 del curso normal de eventos la instrucción encontrada es: "modificar_cadena".<br><br>A2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>A3.- Se posiciona en la lugar específico de la cadena a modificar, estos pueden ser para agregar el nombre de una interface o para insertar o substituir un valor de un atributo.<br><br>A4.- Inserta o modifica valores según corresponda.<br><br>A5.- Regresa al paso 5 del curso normal de eventos. |
| **B.- INSTRUCCIÓN ENCONTRADA (INSERTAR_CÓDIGO)** | |
| | B1.- En el paso 6 del curso normal de eventos la instrucción encontrada es: "insertar_codigo".<br><br>B2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>B3.- Se posiciona en el archivo en el lugar pertinente. |

| | |
|---|---|
| | B4. Escribe el código a insertar.<br><br>B5.- Regresa al paso 5 del curso normal de eventos. |
| **C.- INSTRUCCIÓN ENCONTRADA (TRANSCRIBIR_CÓDIGO)** | |
| | C1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “transcribir_codigo”.<br><br>C2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>C3. Se posiciona en el lugar de las importaciones en el archivo que se está generando.<br><br>C4.- Transcribe las importaciones de la librería al archivo que se está generando.<br><br>C5.- Se posiciona en el lugar de los atributos en el archivo que se está generando.<br><br>C6.- Transcribe los atributos de la librería a la clase que se está generando.<br><br>C7.- Se posiciona en el lugar de insertar métodos de la clase que se está generando<br><br>C8.- Generar métodos Bean (set y get de los atributos) y escribirlos en el archivo que está generando. |

| | |
|---|---|
| | C9.- Regresa al paso 5 del curso normal de eventos. |
| **D.- INSTRUCCIÓN ENCONTRADA (CREAR_INSTANCIA)** | |
| | D1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “crear_instancia”.<br><br>D2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>D3.- Se posiciona en el archivo en el lugar pertinente.<br><br>D4.- Genera la línea de código para la instancia.<br><br>D5.- Escribe la línea de código en el archivo que se está generando.<br><br>D6.- Regresa al paso 5 del curso normal de eventos. |
| **E.- INSTRUCCIÓN ENCONTRADA (LLAMAR_MÉTODO)** | |
| | E1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “llamar_metodo”.<br><br>E2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>E3.- Se posiciona en el archivo en el lugar pertinente. |

| | |
|---|---|
| | E4.- Genera la línea de código a insertar.<br><br>E5.- Escribe la línea de código en el archivo que se está generando.<br><br>E6.- Regresa al paso 5 del curso normal de eventos. |
| **F.- INSTRUCCIÓN ENCONTRADA (TRANSCRIBIR_CLASE)** | |
| | F1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “transcribir_clase”.<br><br>F2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>F3.- Crea el archivo .java haciendo una copia fiel de la librería.<br><br>F4.- Regresa al paso 5 del curso normal de eventos. |
| **G.- INSTRUCCIÓN ENCONTRADA (GUARDAR_FIGURA)** | |
| | G1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “guardar_figura”.<br><br>G2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>G3.- Crea Archivo .java de la figura.<br><br>G4.- Se posiciona en el lugar pertinente del archivo .java de la figura que se está |

| | |
|---|---|
| | generando.<br><br>G5.- Escribe las Instrucciones para construir la figura.<br><br>G6.- Regresa al paso 5 del curso normal de eventos. |
| **H.- INSTRUCCIÓN ENCONTRADA (AGREGAR_INTERFACE)** | |
| | H1.- En el paso 5 del curso normal de eventos la instrucción encontrada es: “agregar_interface”.<br><br>H2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.<br><br>H3.- Se posiciona en el inicio del archivo que se está generando para agregar el nombre de la interface.<br><br>H4.- inserta el nombre de la interface.<br><br>H5.- Se posiciona en el lugar donde se puede insertar métodos.<br><br>H6.- Se Transcribe el código de la librería a la clase que se está generando.<br><br>H7.- Regresa al paso 5 del curso normal de eventos. |
| **I.- INSTRUCCIÓN NO ENCONTRADA.-** | |
| | I1.- En el paso 5 del Curso normal de Eventos no se encuentra ninguna instrucción en el |

| | |
|---|---|
| | vector de instrucciones.<br><br>I2.- Regresa al paso 6 del Curso Normal de Eventos. |

## USE CASE GENERAR CÓDIGO

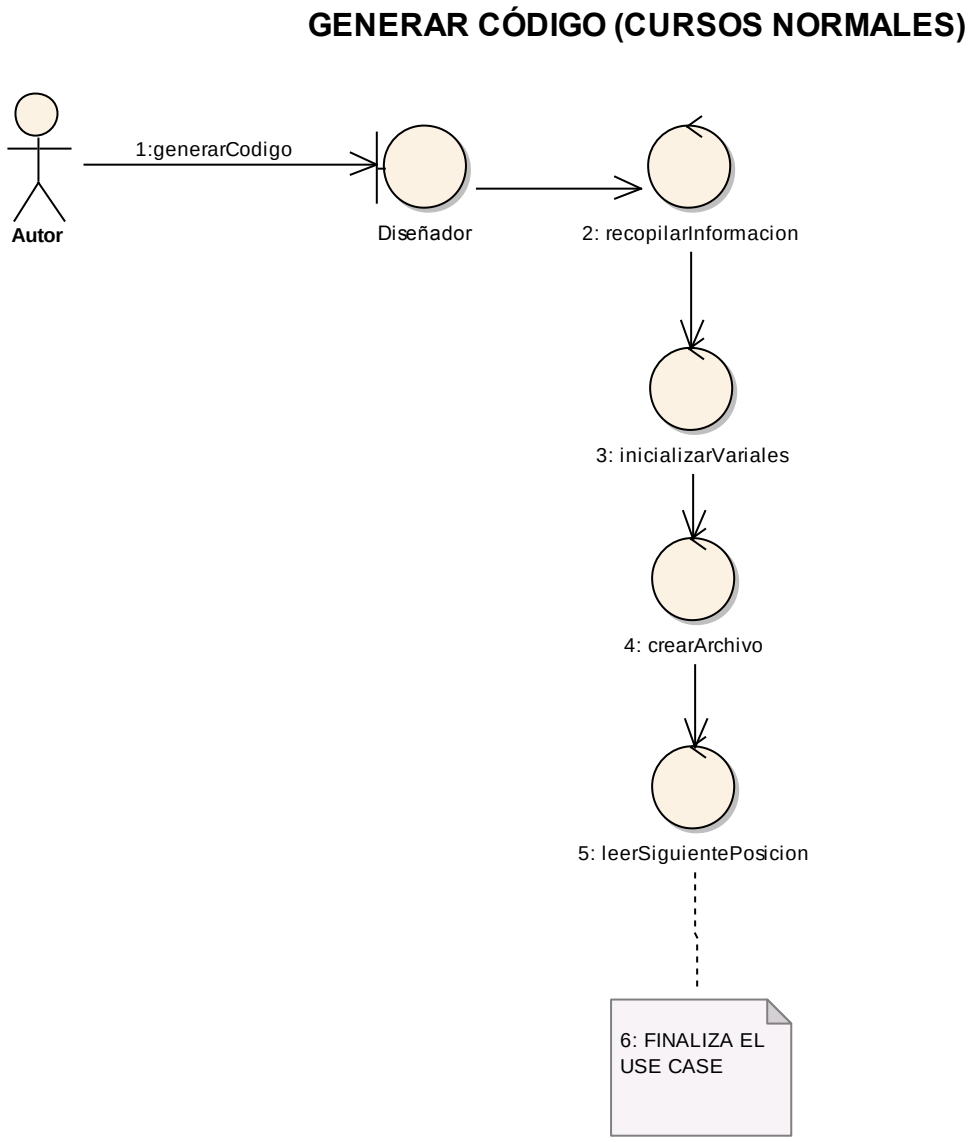



Figura 56.- Curso Normal: Generar Código.

**GENERAR CÓDIGO (CURSOS ALTERNOS)**

**CURSO ALTERNO A: INSTRUCCION ENCONTRADA(MODIFICAR CADENA)**

A: INSTRUCCION ENCONTRADA "modificar_cadena" (Viene del paso 5 del Curso Normal de Eventos)

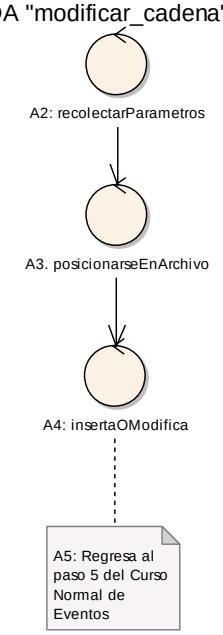



Figura 57.- Curso Alterno A: Instrucción Encontrada (Modificar Cadena).

**CURSO ALTERNO B: INSTRUCCION ENCONTRADA(INSERTAR CODIGO)**

B: INSTRUCCION ENCONTRADA "insertar_codigo" (Viene del paso 5 del Curso Normal de Eventos)

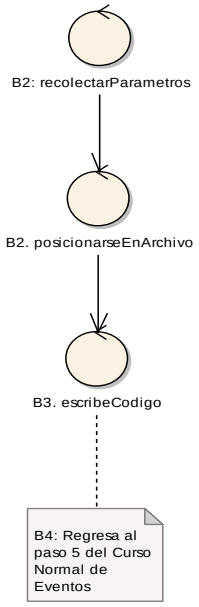


Figura 58.- Curso Alterno B: Instrucción Encontrada (Insertar Código).

**CURSO ALTERNO C: INSTRUCCION ENCONTRADA(TRANSCRIBIR CODIGO)**

C: INSTRUCCION ENCONTRADA "transcribir_codigo" (Viene del paso 5 del Curso Normal de Eventos)

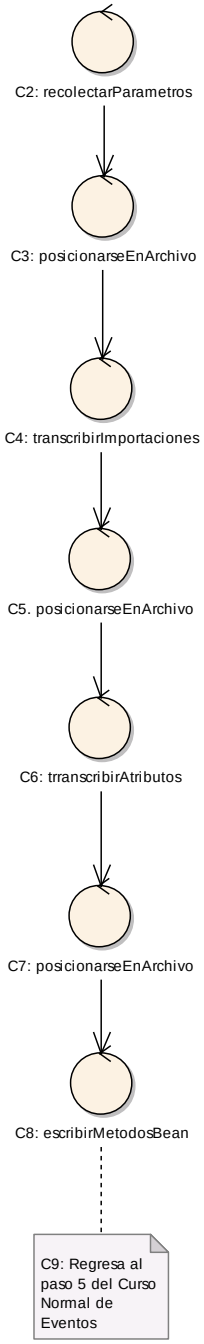



Figura 59.- Curso Alterno C: Transcribir Código.

**CURSO ALTERNO D: INSTRUCCION ENCONTRADA(CREAR INSTANCIA)**

D: INSTRUCCION ENCONTRADA "crear_instancia" (Viene del paso 5 del Curso Normal de Eventos)

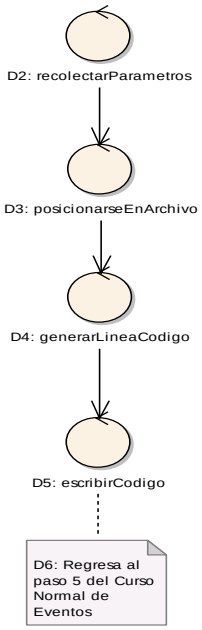



Figura 60.- Curso Alterno D: Crear Instancia.

**CURSO ALTERNO E: INSTRUCCION ENCONTRADA(LLAMAR METODO)**

E: INSTRUCCION ENCONTRADA "llamar_metodo" (Viene del paso 5 del Curso Normal de Eventos)

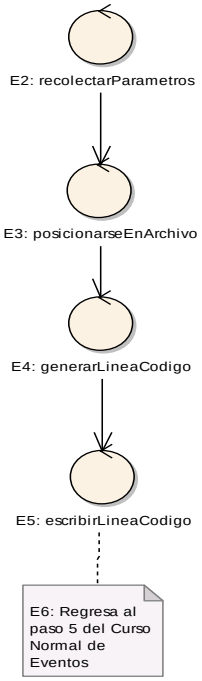


Figura 61.- Curso Alterno E: Llamar Método.

## CURSO ALTERNO F: INSTRUCCION ENCONTRADA(TRANSCRIBIR CLASE)

F: INSTRUCCION ENCONTRADA "transcribir_clase" (Viene del paso 5 del Curso Normal de Eventos)

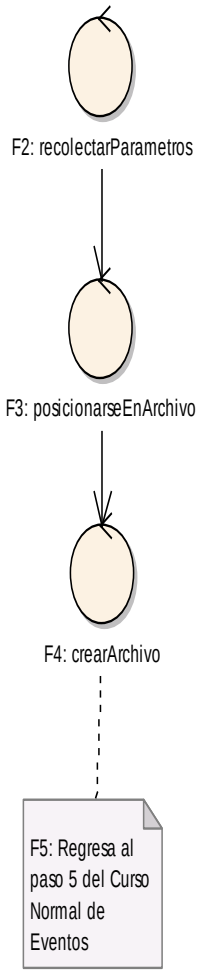



Figura 62.- Curso Alterno F: Transcribir Clase

## CURSO ALTERNO G: INSTRUCCION ENCONTRADA(GUARDAR FIGURA)

G: INSTRUCCION ENCONTRADA "guardar_figura" (Viene del paso 5 del Curso Normal de Eventos)

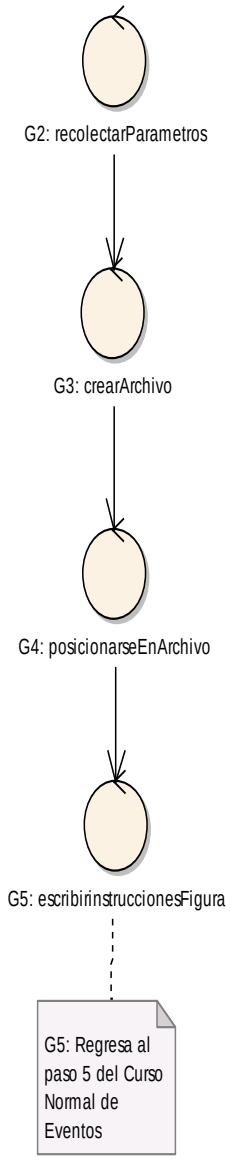



Figura 63.- Curso Alterno G: Guardar Figura.

**CURSO ALTERNO H: INSTRUCCION ENCONTRADA(AGREGAR INTERFAZ)**

H: INSTRUCCION ENCONTRADA "agregra_interfaz" (Viene del paso 5 del Curso Normal de Eventos)

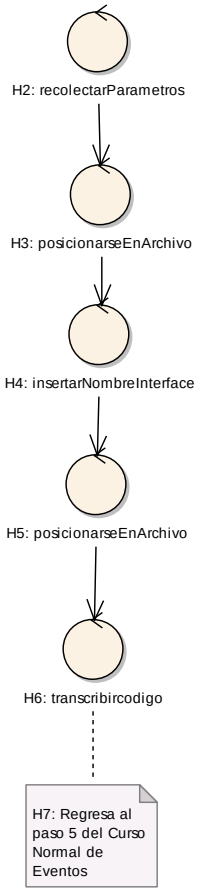



Figura 64.- Curso Alterno H: Agregar Interface.

**CURSO ALTERNO I: INSTRUCCION NO ENCONTRADA**

I: INSTRUCCION ENCONTRADA "instruccion_no_encontrada" (Viene del paso 5 del Curso Normal de Eventos)

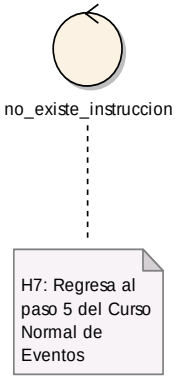



Figura 64(A).- Curso Alterno I: Instrucción no Encontrada.

DIAGRAMA DE SECUENCIA: GENERAR CODIGO

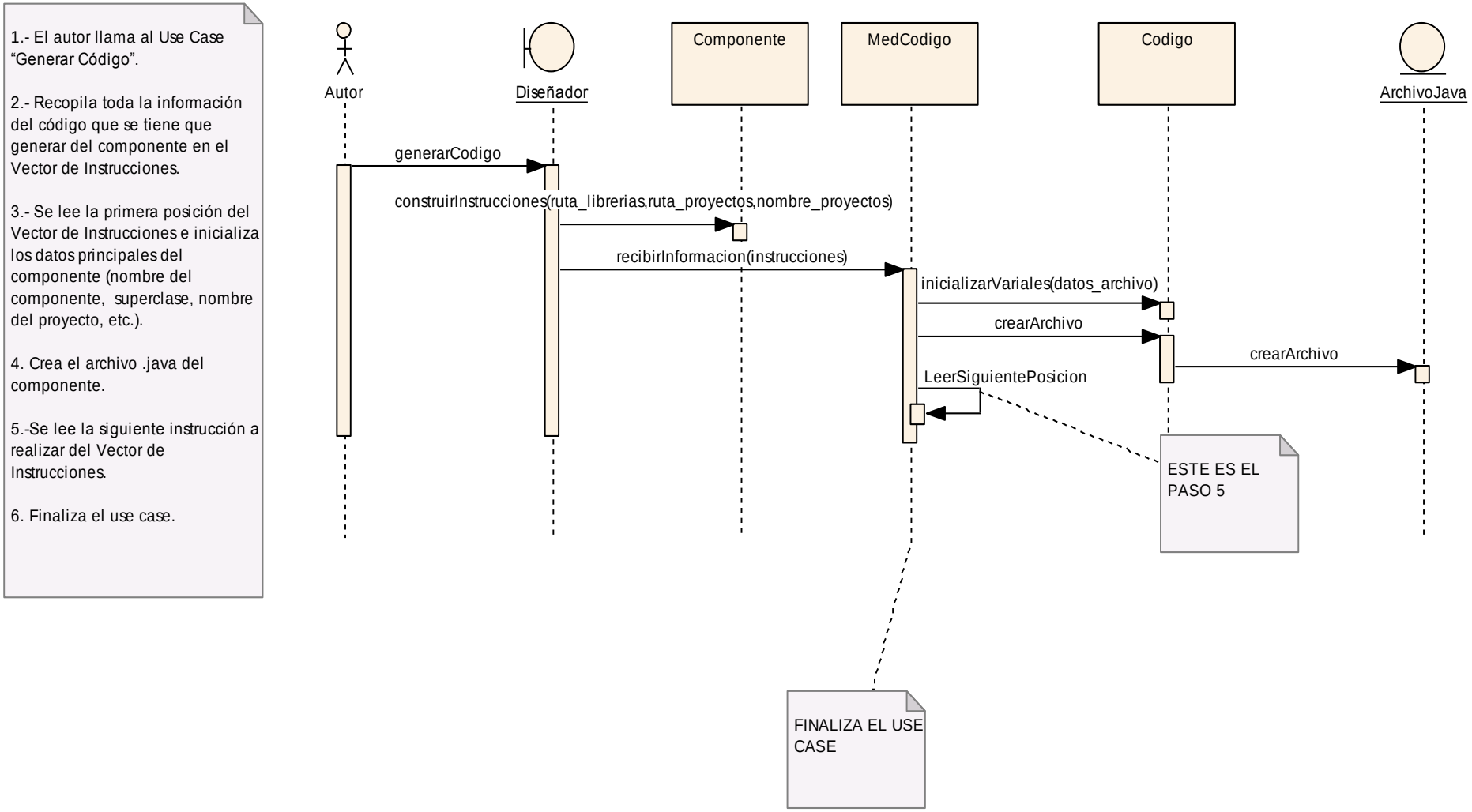


Figura 65.- Diagrama de Secuencia Generar Código.

CURSO ALTERNO A: INSTRUCCIÓN ENCONTRADA: modicar_cadena

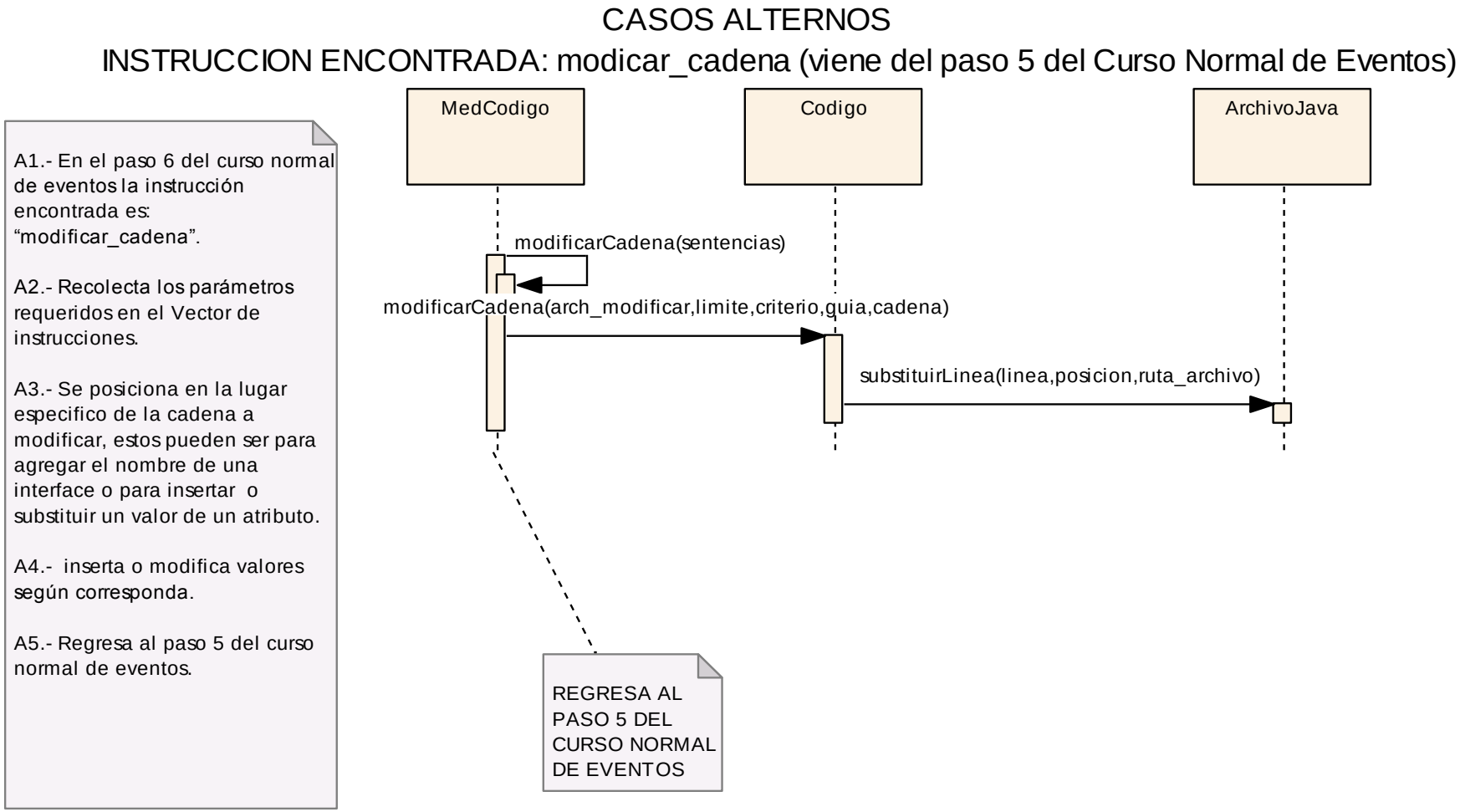



Figura 66.- Diagrama de Secuencia Curso Alterno A: Modificar Cadena.

CURSO ALTERNO B INSTRUCCIÓN ENCONTRADA: insertar_codigo

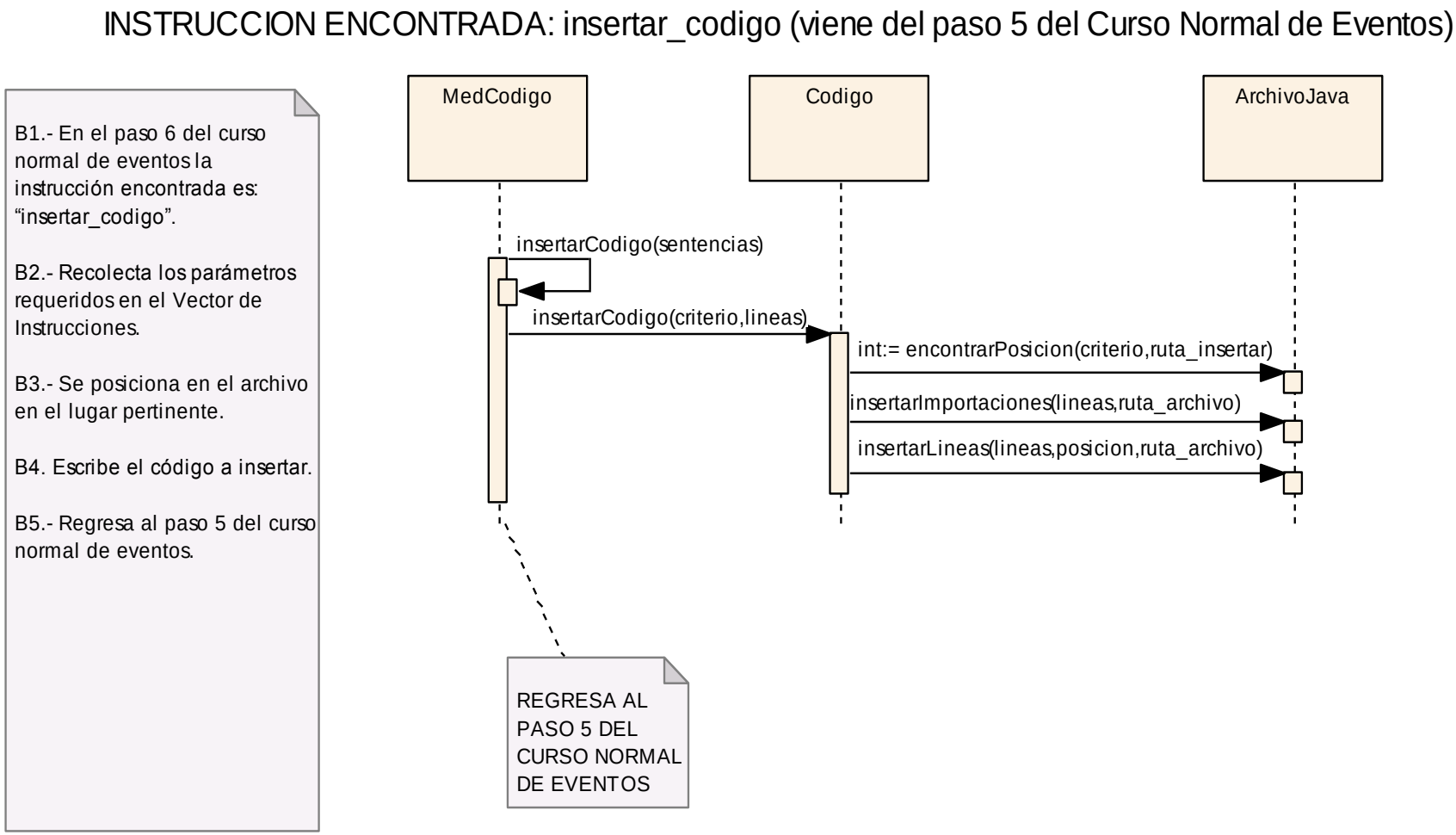



Figura 67.- Diagrama de Secuencia Curso Alterno B: Insertar Código.

CURSO ALTERNO C: INSTRUCCIÓN ENCONTRADA: transcribir_codigo

INSTRUCCION ENCONTRADA: transcribir_codigo (viene del paso 5 del Curso Normal de Eventos)

C1.- En el paso 6 del curso normal de eventos la instrucción encontrada es: “transcribir_codigo”.

C2.- Recolecta los parámetros requeridos en el Vector de Instrucciones.

C3. Se posiciona en el lugar de las importaciones en el archivo que se está generando.

C4.- Transcribe las importaciones de la librería al archivo que se está generando.

C5.- Se posiciona en el lugar de los atributos en el archivo que se está generando.

C6.- Transcribe los atributos de la librería a la clase que se está generando.

C7.- Se posiciona en el lugar de insertar métodos de la clase que se está generando

C8.- Generar métodos Bean (set y get de los atributos) y escribirlos en el archivo que está generando.

C9.- Regresa al paso 5 del curso normal de eventos.

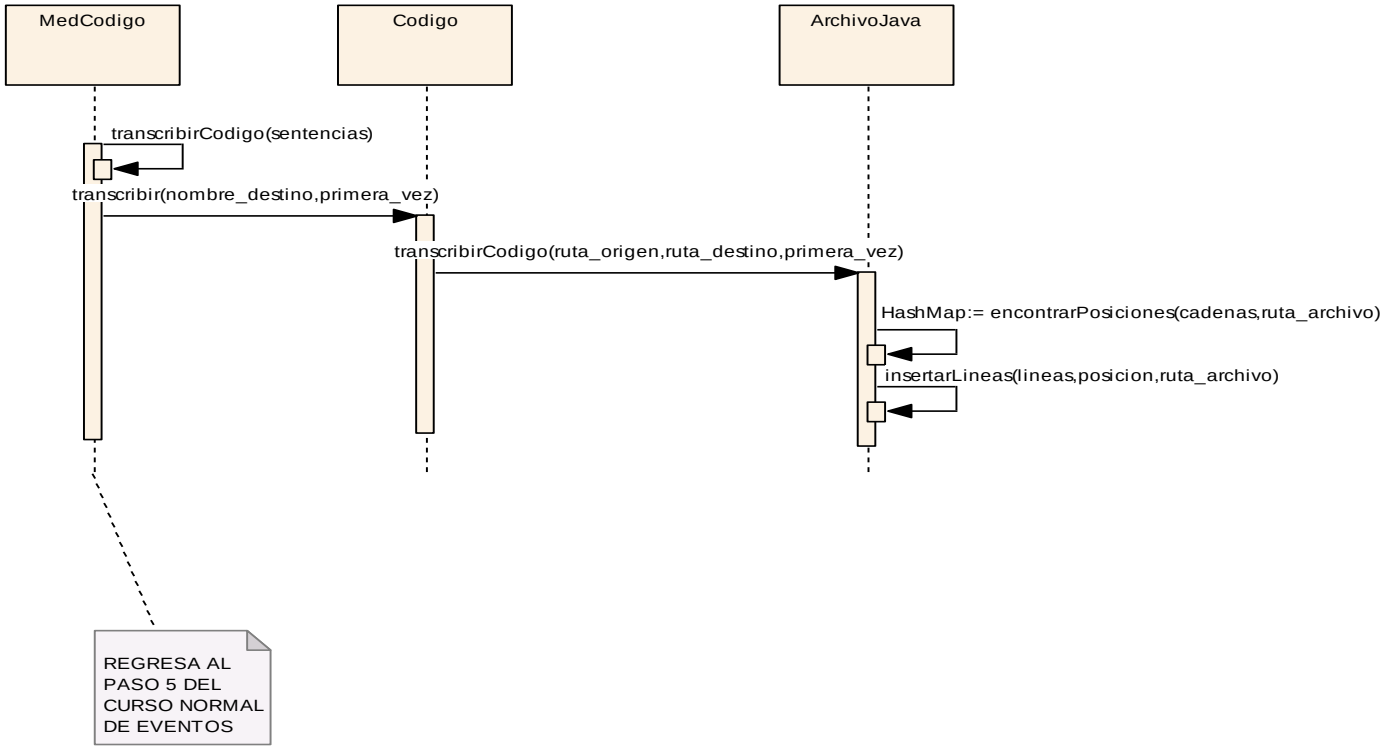


Figura 68.- Diagrama de Secuencia Curso Alterno C: Transcribir Código

CURSO ALTERNO D: INSTRUCCIÓN ENCONTRADA: crear_instancia

INSTRUCCION ENCONTRADA: crear_instancia(viene del paso 5 del Curso Normal de Eventos)

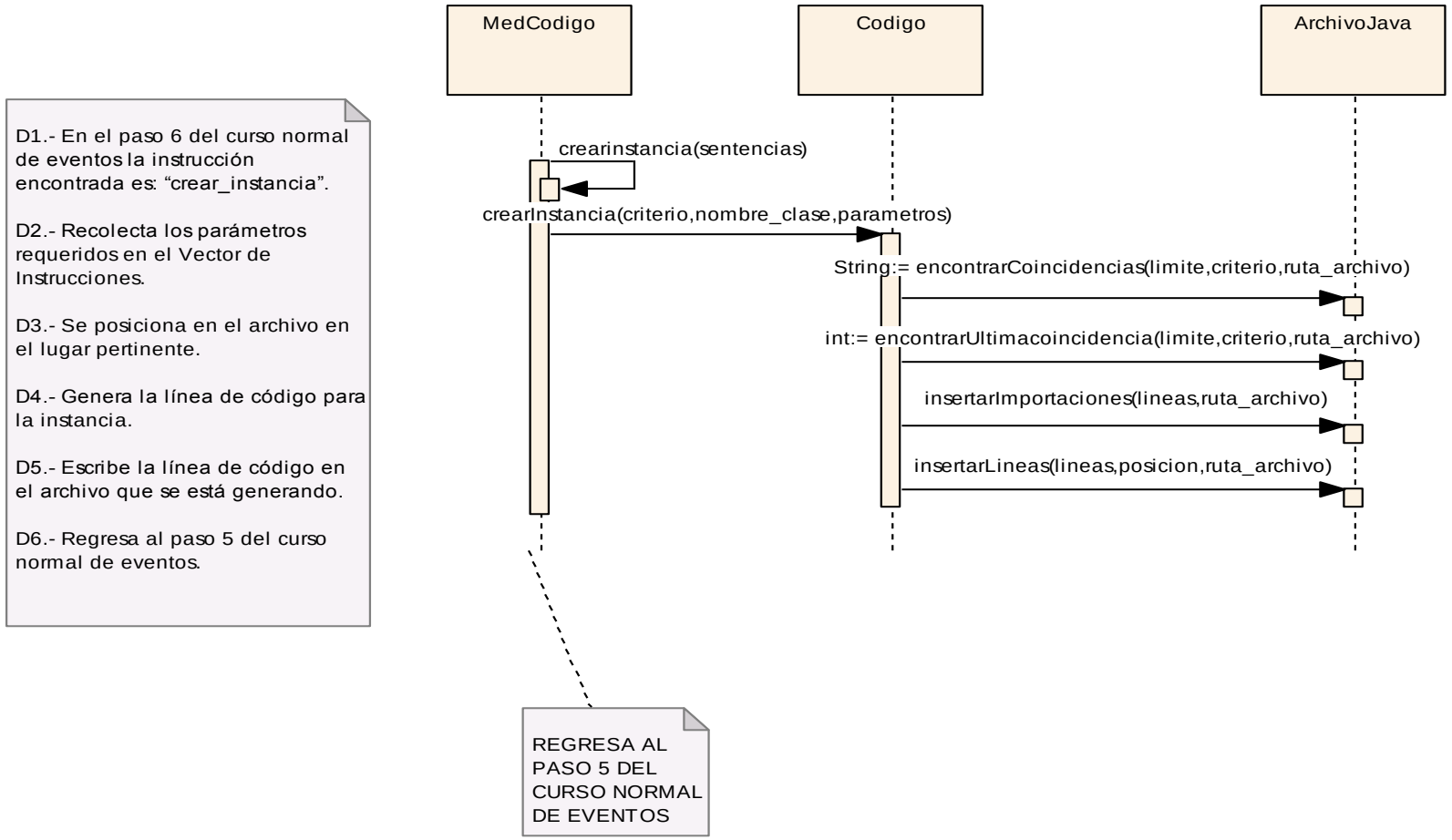


Figura 69.- Diagrama de Secuencia Curso Alterno D: Crear Instancia

CURSO ALTERNO E: INSTRUCCIÓN ENCONTRADA: llamar_metodo

INSTRUCCION ENCONTRADA: llamar_metodo (viene del paso 5 del Curso Normal de Eventos)

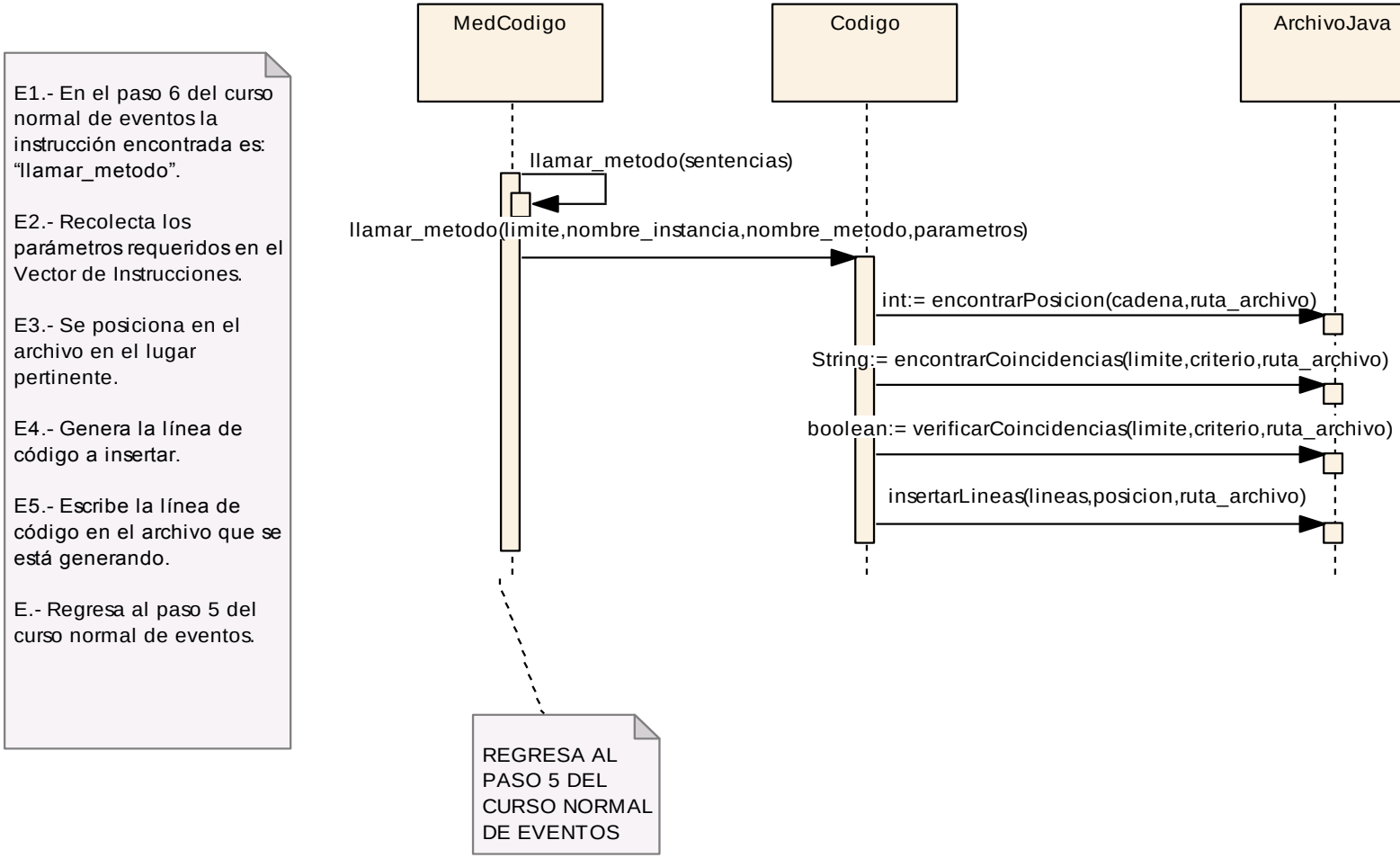



Figura 70.- Diagrama de Secuencia  Curso Alterno E: Llamar Método.

CURSO ALTERNO F: INSTRUCCIÓN ENCONTRADA: transcribir_clase

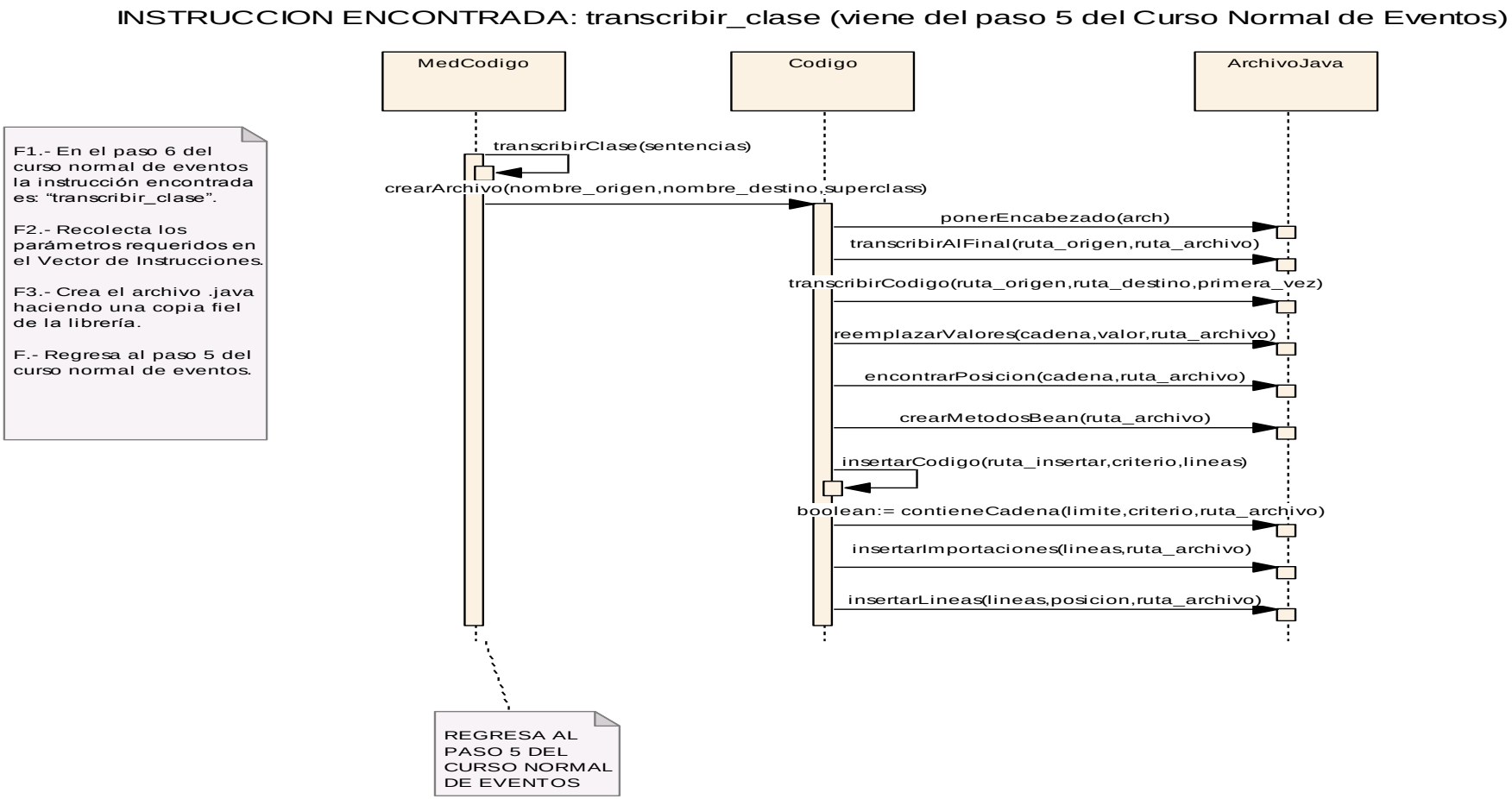


Figura 71.- Diagrama de Secuencia Curso Alterno F: Transcribir Clase.

CURSO ALTERNO G: INSTRUCCIÓN ENCONTRADA: guardar_figura

INSTRUCCION ENCONTRADA: guardar_figura (viene del paso 5 del Curso Normal de Eventos)

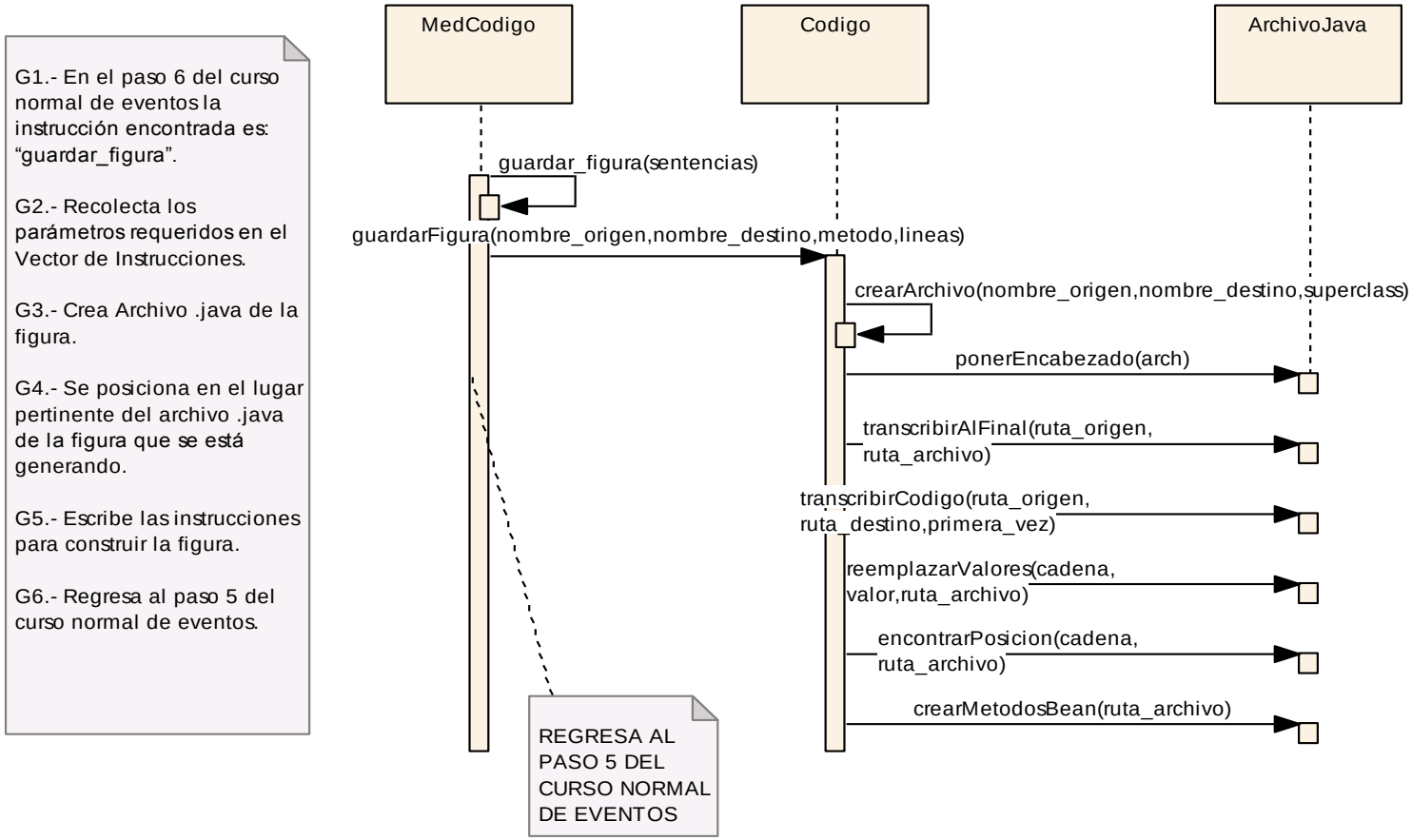


Figura 72.- Diagrama de Secuencia Curso Alterno G: Guardar Figura.

CURSO ALTERNO H: INSTRUCCIÓN ENCONTRADA: agregar_interface

INSTRUCCION ENCONTRADA: agregar_interface(viene del paso 5 del Curso Normal de Eventos)

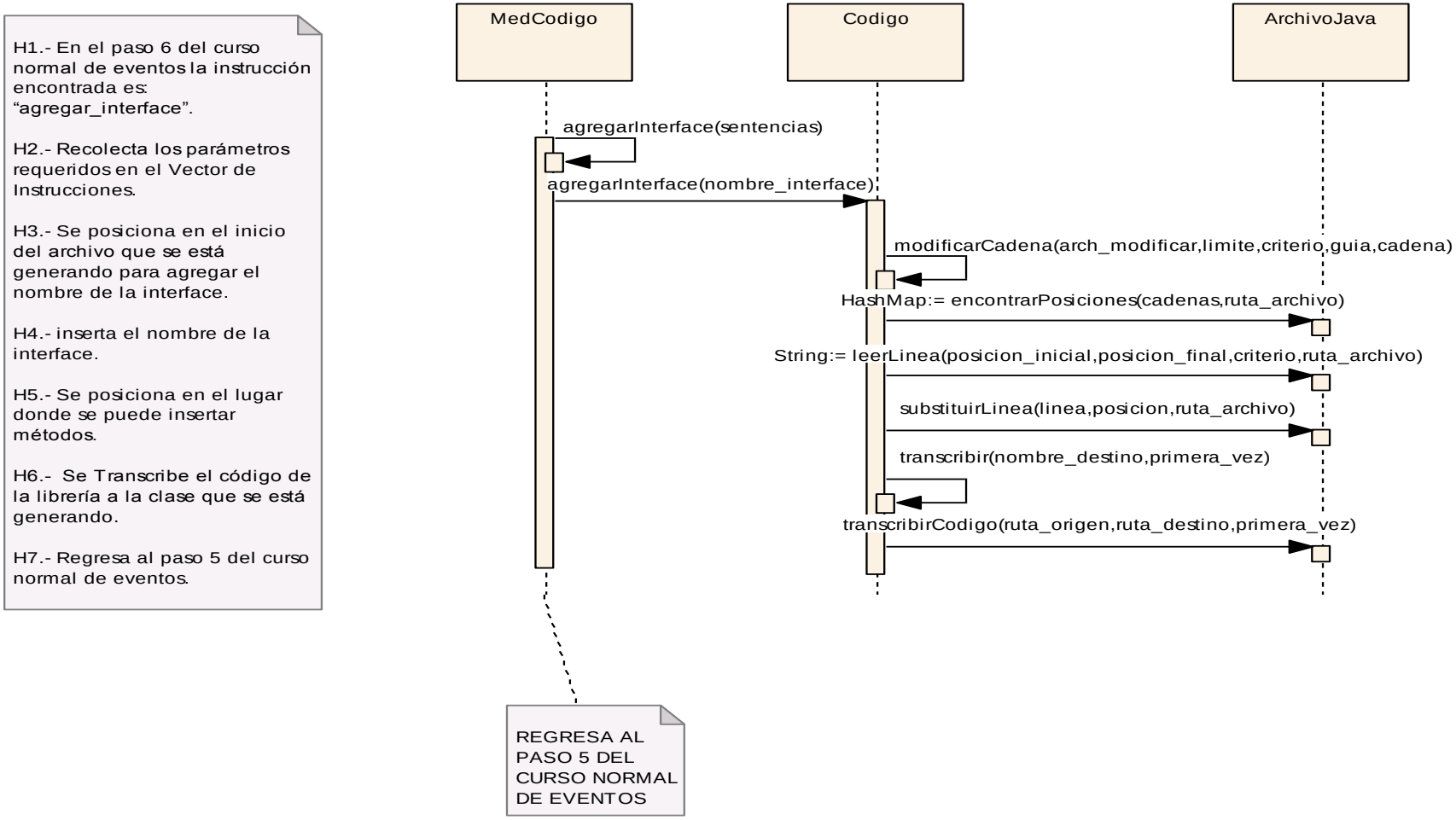


Figura 73.- Diagrama de Secuencia Curso Alterno H: Agregar Interface.

## 6.4. MODELAMIENTO ESTÁTICO.-

### 6.4.1. Diagrama de Paquetes.-

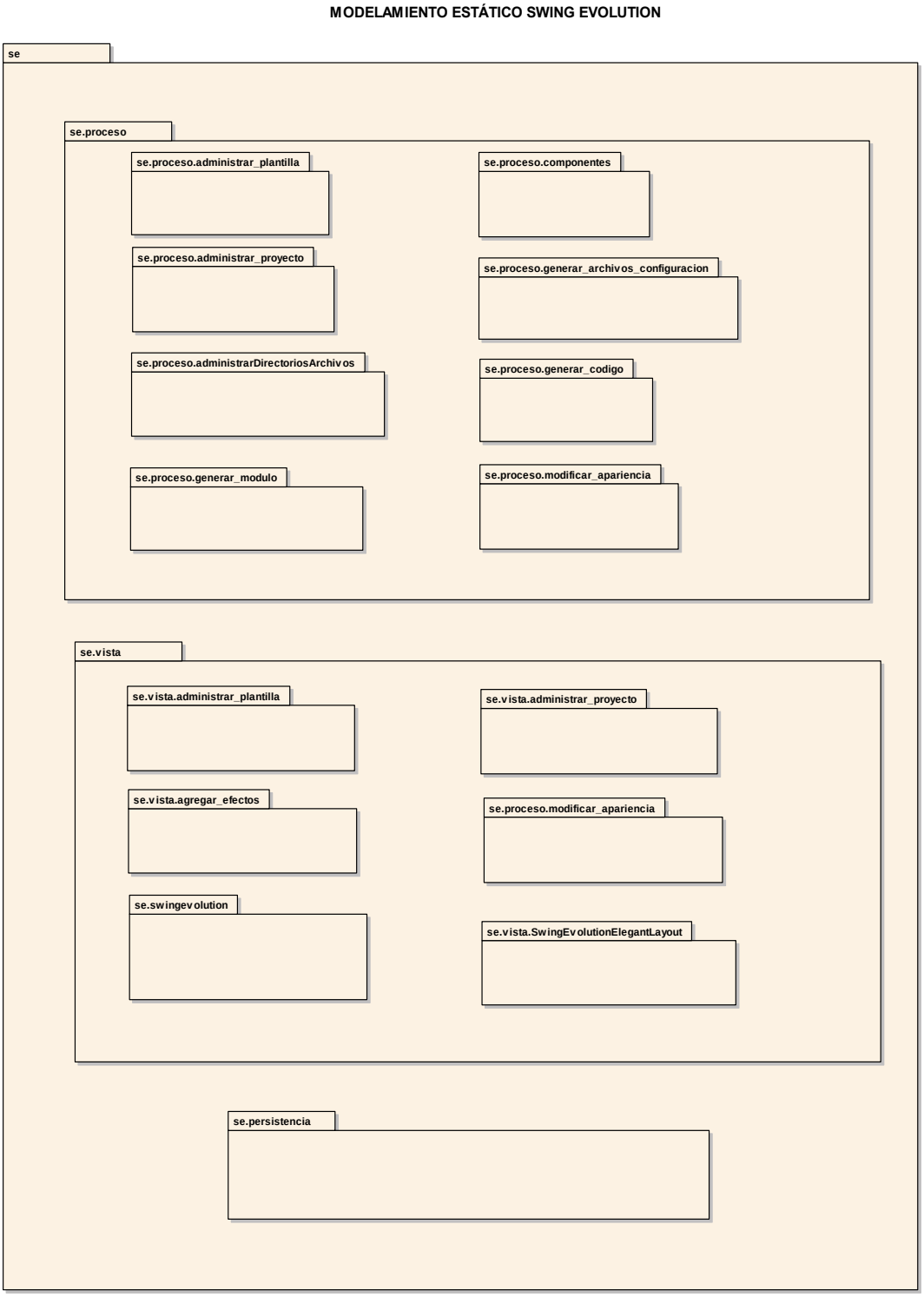



Figura 74.- Diagrama de Paquetes.

## 6.4.2. Diagrama de Clases por cada use case

### Use Case: Administrar Proyecto

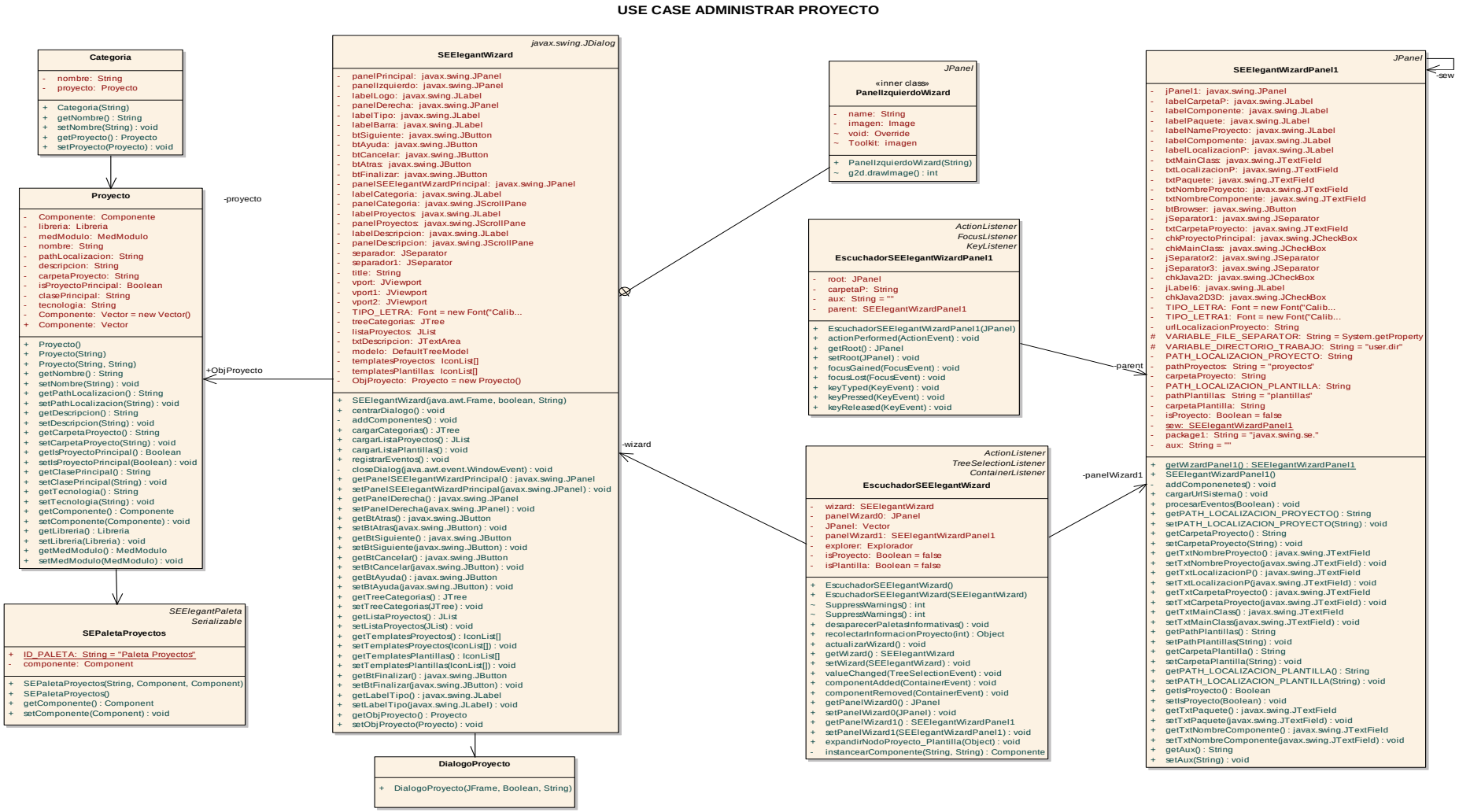



Figura 75.- Diagrama Clases Use Case Administrar Proyecto.

## Use Case: Administrar Plantilla

### USE CASE ADMINISTRAR PLANTILLA

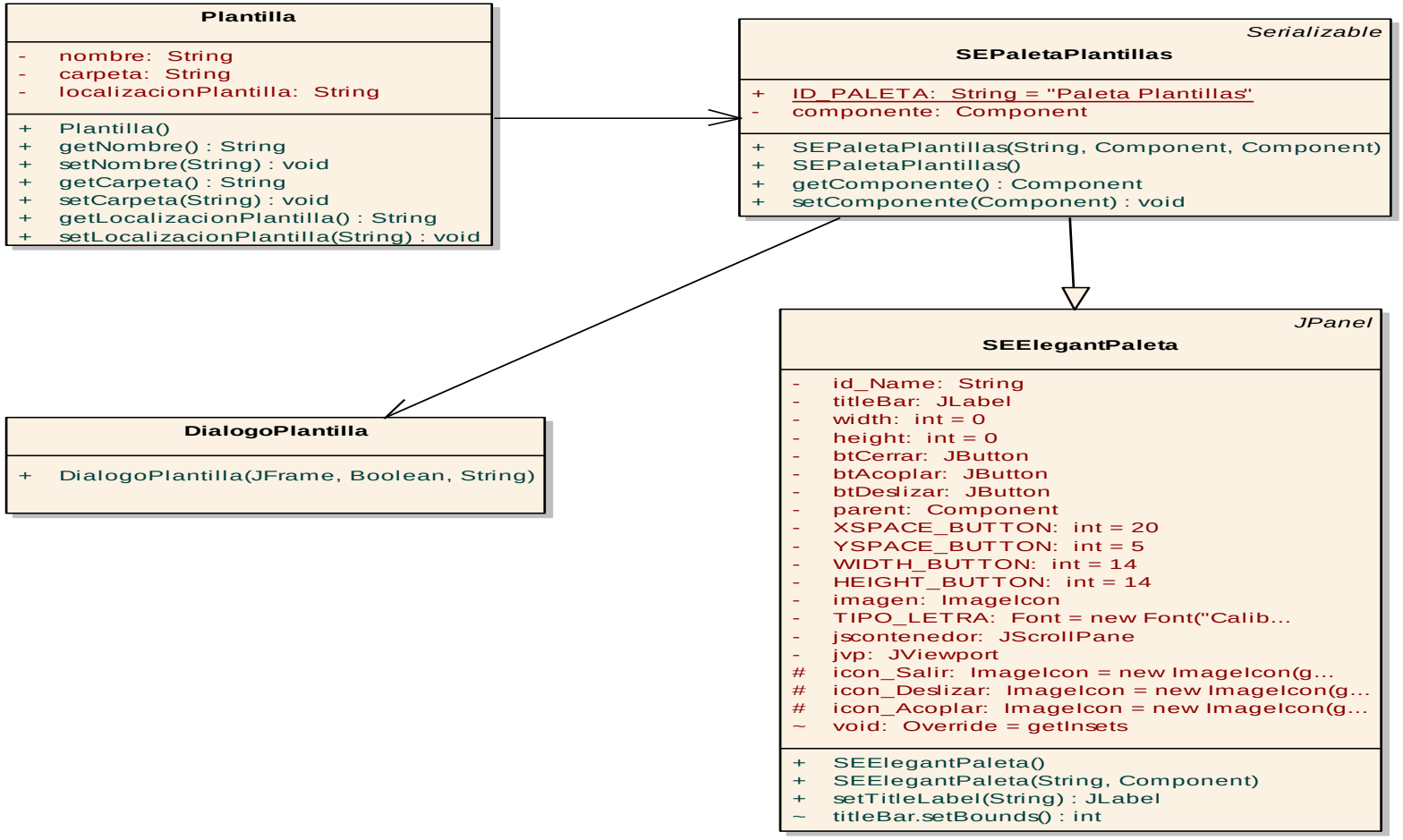


Figura 76.- Diagrama Clases Use Case Administrar Plantilla.

# Use Case: Modificar Apariencia

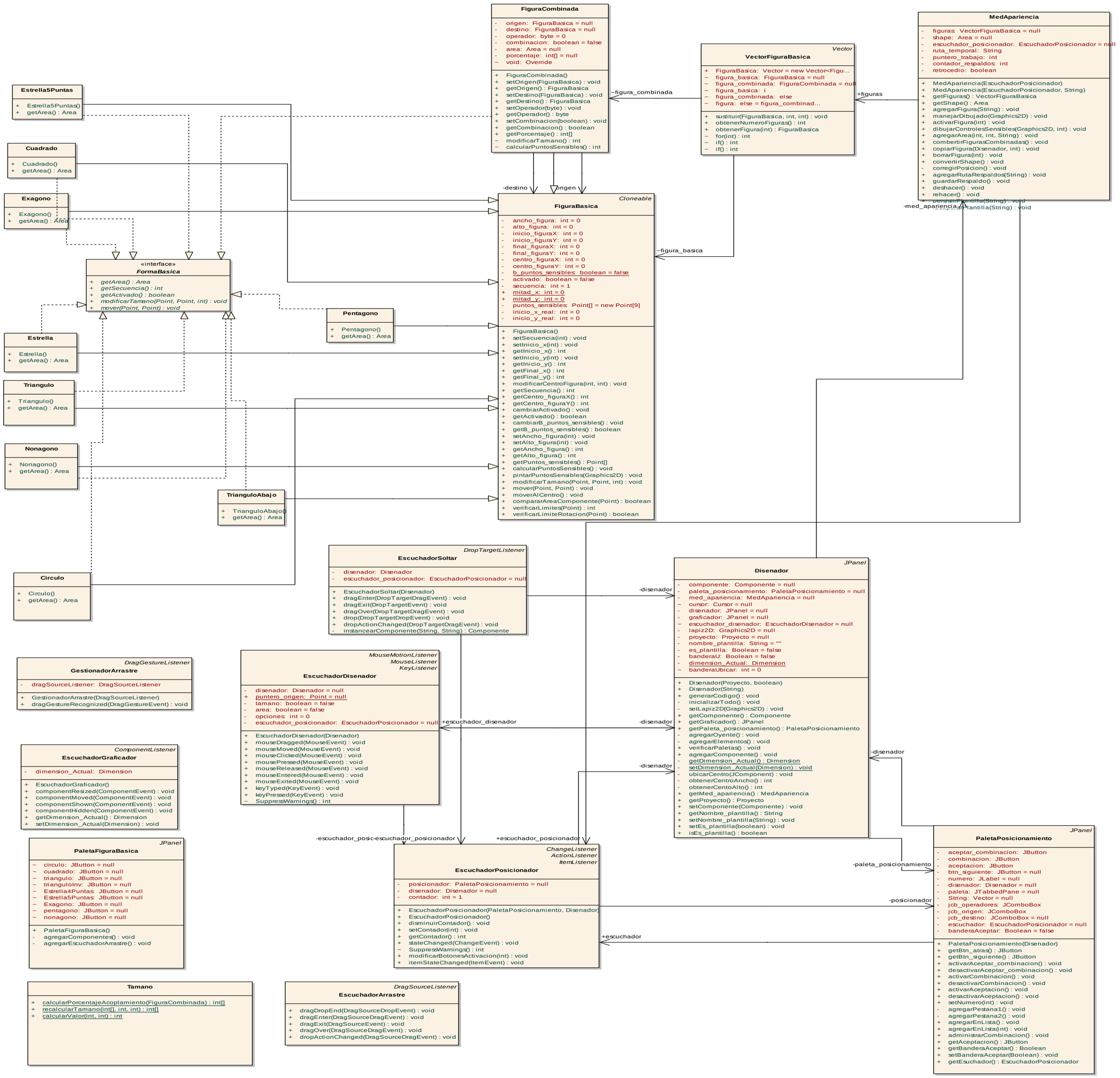


Figura 77.- Diagrama Clases Use Case Modificar Apariencia

# Use Case: Agregar Efectos

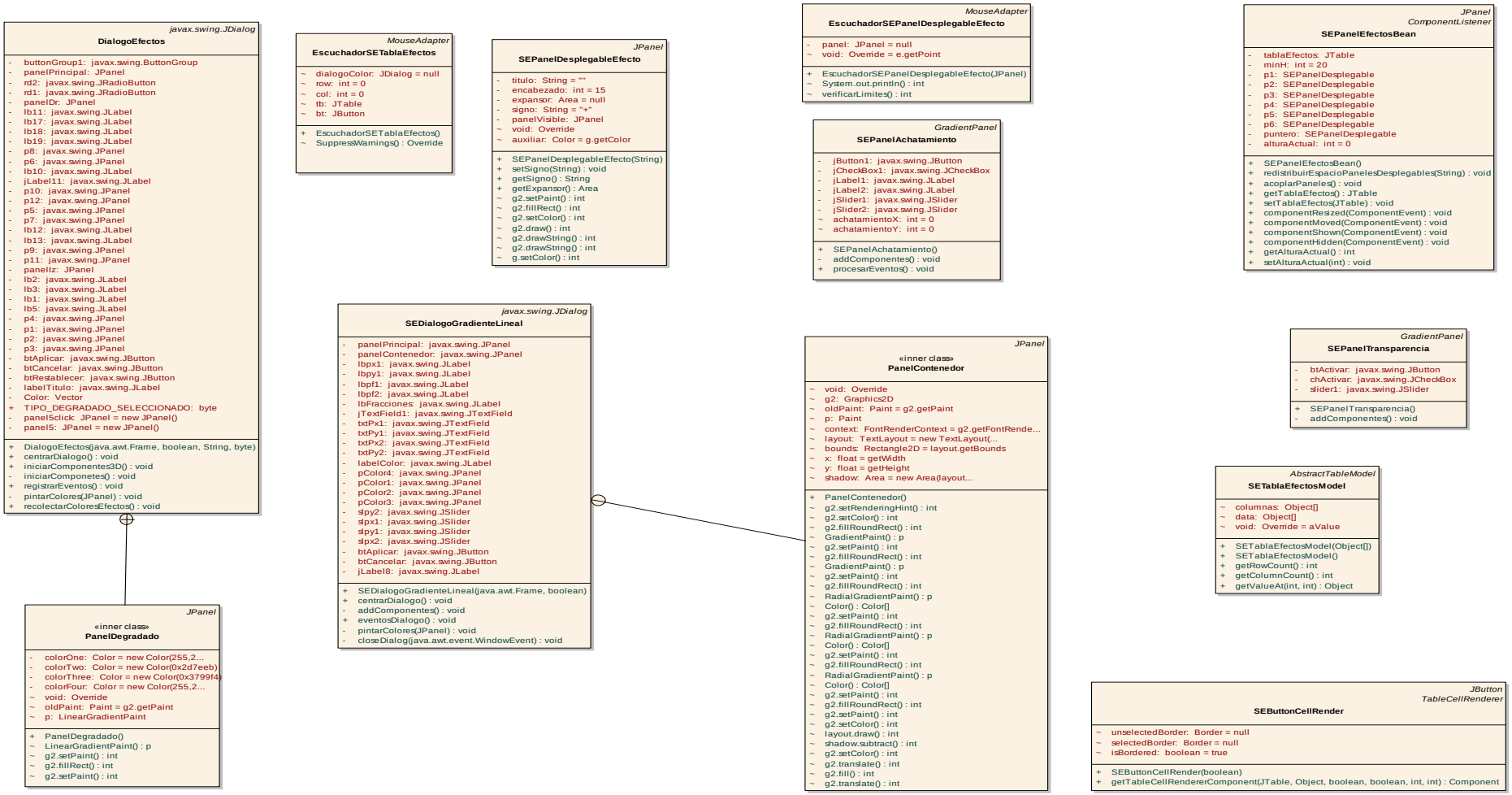


Figura 78.- Diagrama de Clases  Use Case Agregar Efectos.

## Use Case: Generar Código

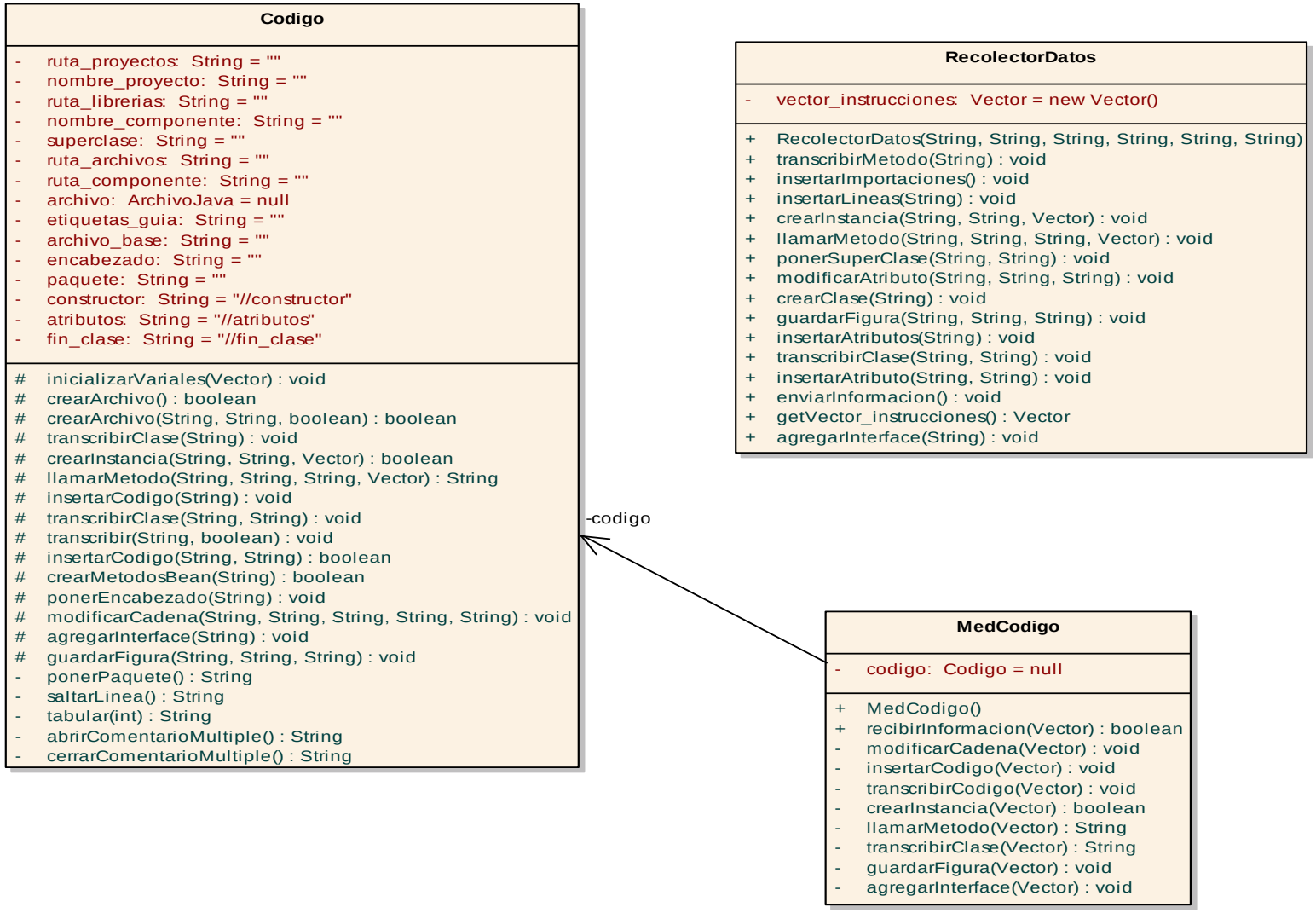


Figura 79.- Diagrama de Clases Use Case Generar Código.

**Use Case: Generar Modulo**

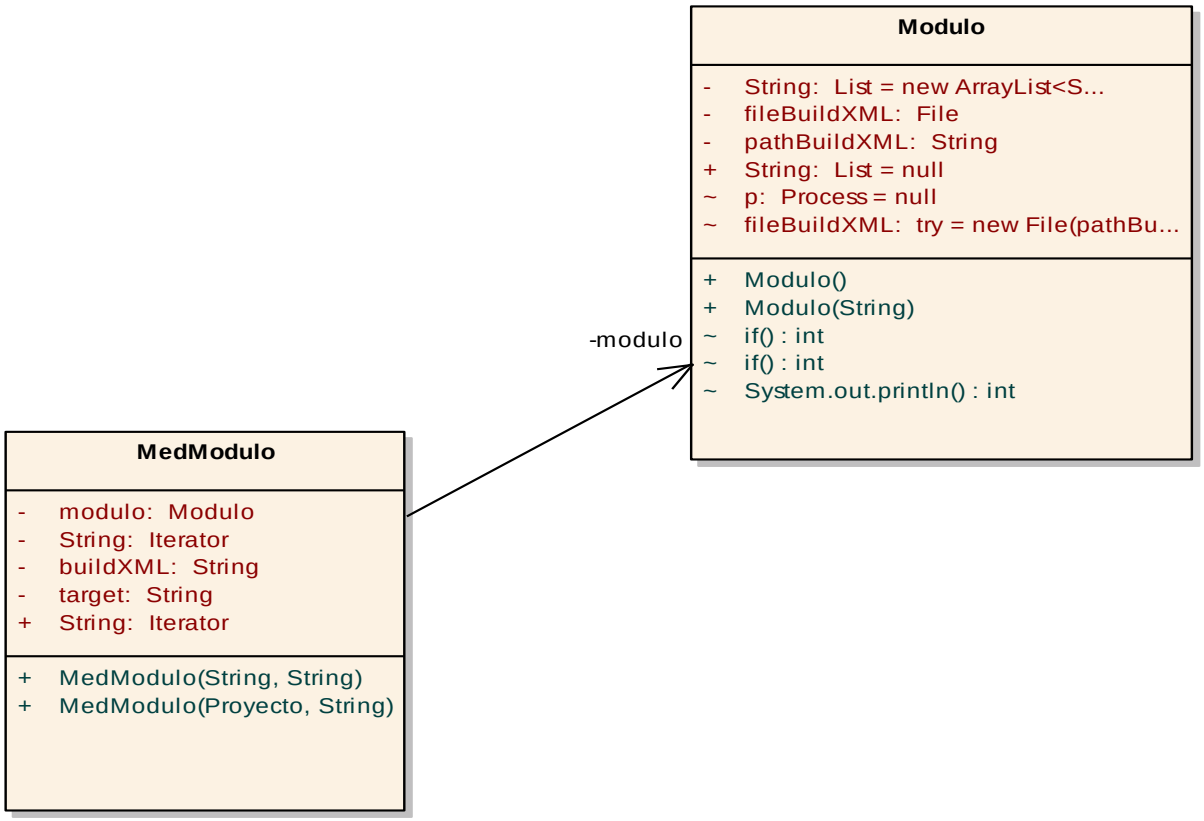



Figura 80.- Diagrama de Clases Use Case Generar Módulo.

## Use Case: Generar Archivos Configuración

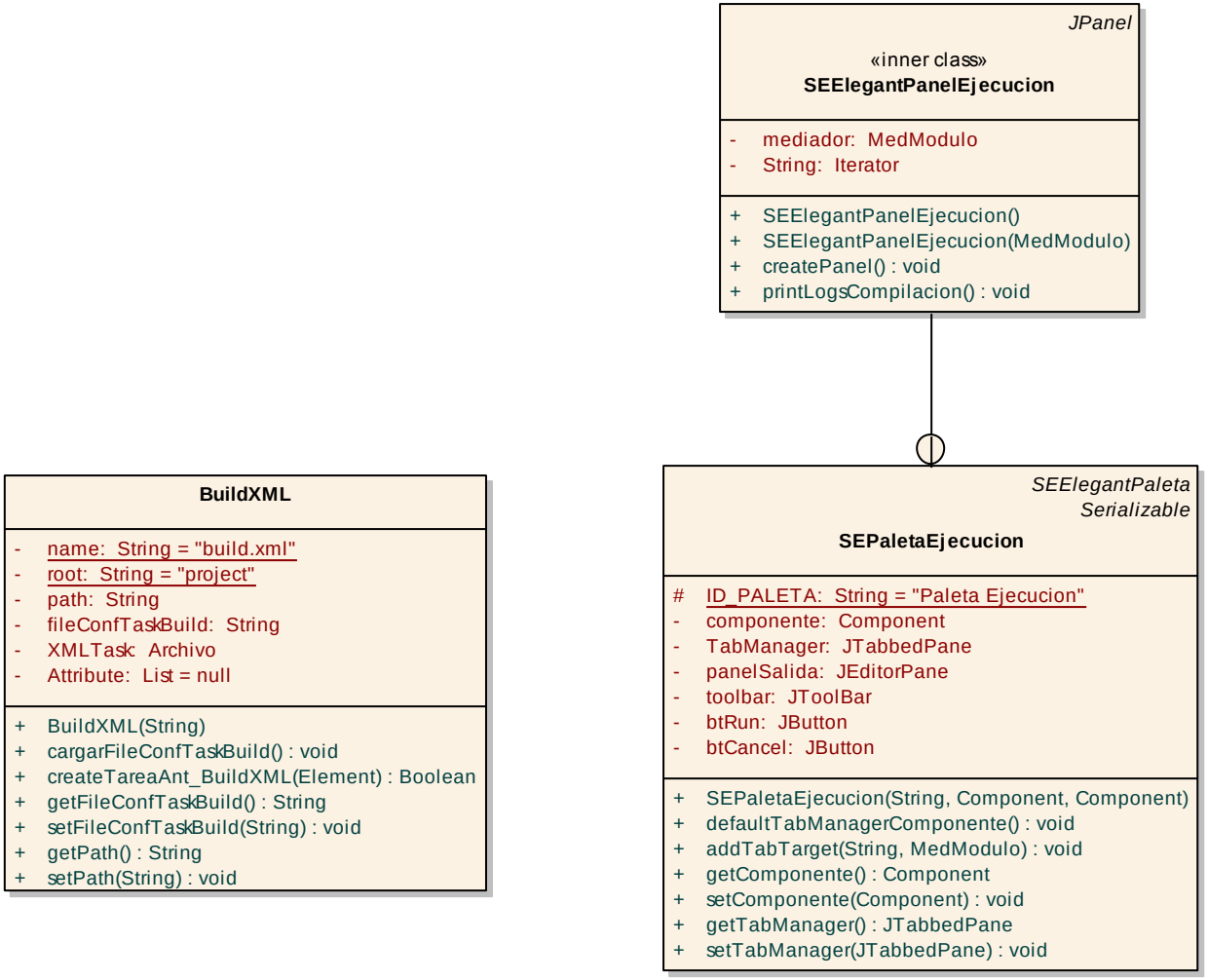
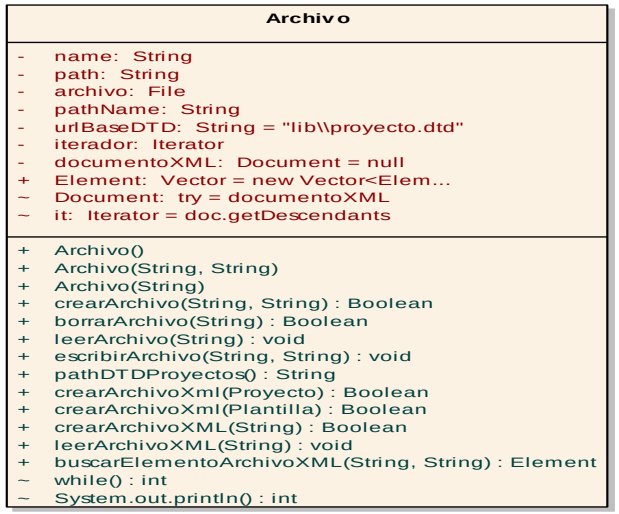
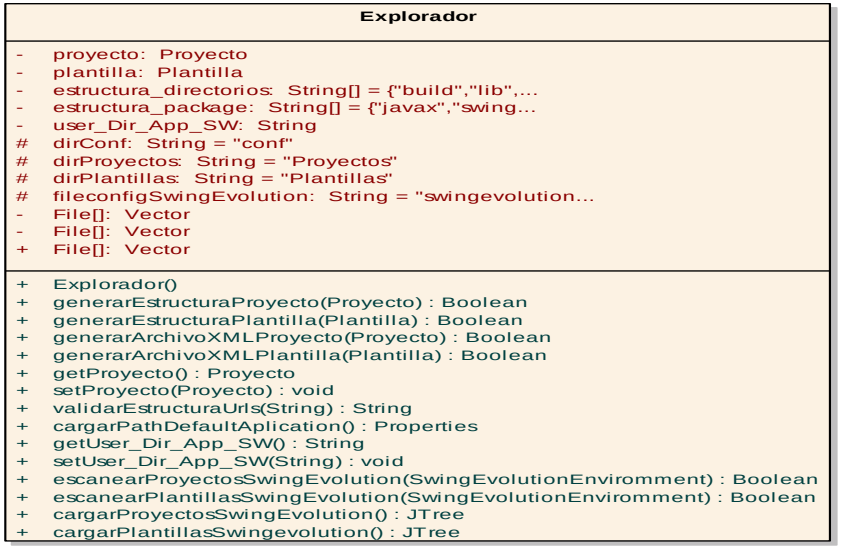


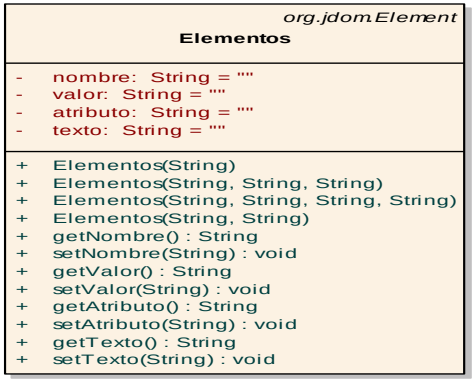
Figura 81.- Diagrama de Clases Use Case Generar Archivos Configuración.

## Use Case: Administrar Directorios y Archivos







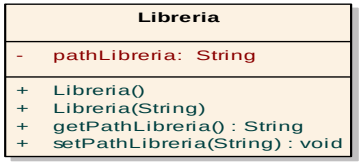



Figura 82.- Diagrama de Clases Use Case Administrar Directorios Archivos.

### 6.4.3. Modelo y Arquitectura.-

#### 6.4.3.1. Diagrama de Componentes.-

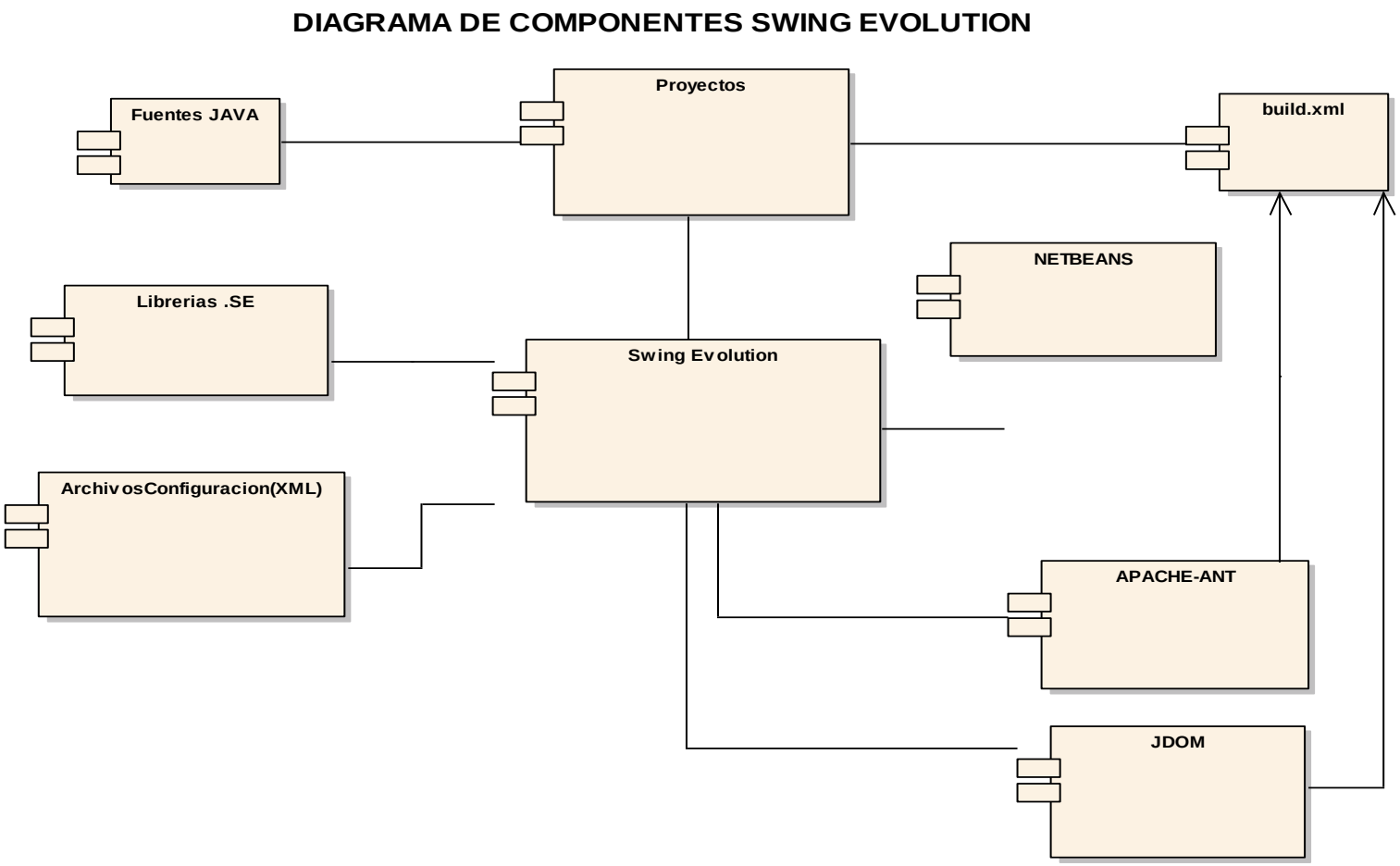



Figura 83.- Diagrama de Componentes Swing Evolution.

## 6.5. PRUEBAS Y VALIDACIÓN.-

La fase de pruebas y validación de la aplicación se llevo a cabo el día 18 de Marzo del 2009. Las pruebas fueron aplicadas a los docentes y estudiantes de la Carrera de Ingeniería en Sistemas en el Área de Energía, las Industrias y los Recursos Naturales No Renovables, cumpliendo de esta manera con uno de los objetivos del proyecto.

Las encuestas fueron realizadas teniendo en cuenta las siguientes categorías: fácil manejo de la aplicación, organización de las pantallas, componentes generados, ambiente amigable al usuario, rendimiento de la aplicación entre otros.

Dentro de este proceso de validación de la aplicación Swing Evolution las pruebas fueron aplicadas a 3 ingenieros de la carrera, 2 responsables del proyecto Swing Evolution y 22 estudiantes de la carrera de Ingeniería en sistemas los cuales harán uso de esta aplicación, siendo un total de 27 encuestas realizadas.

Durante la aplicación de las pruebas no se encontraron errores de ejecución, se hicieron observaciones en cuanto al rendimiento del sistema, lo que se tomó en cuenta para validar y encontrar errores que pudieran suceder.

Al final de la pruebas se concluyó que la Aplicación **"DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA VISUAL QUE PERMITA LA CREACIÓN DE NUEVOS COMPONENTES SWING UTILIZANDO JAVA2D Y 3D"** cumple con todos los requerimientos planteados al inicio del proyecto.

A continuación se detalla los resultados de la tabulación de las pruebas de **FUNCIONALIDAD Y RENDIMIENTO** de la aplicación:

**VARIABLES DE CALIFICACIÓN.-**

- **EX = Excelente**
- **MB = Muy Buena**
- **B = Buena**
- **R =Regular**

**DOCENTES Y ESTUDIANTES**

| FUNCIONALIDAD | Ex | MB | B | R | TOTAL |
|---|---|---|---|---|---|
| Cuál sería su calificación acerca del Ambiente de Desarrollo Swing Evolution 1.1.0 en lo que corresponde a su Entorno Gráfico. | 17 | 9 | 1 | 0 | 27 |
| La aplicación le parece amigable al usuario. | 27 | 0 | 0 | 0 | 27 |
| Qué tal le parece la distribución de los diferentes controles visuales dentro de la aplicación Swing Evolution 1.1.0 | 9 | 18 | 0 | 0 | 27 |
| Cómo calificaría a cada uno de los componentes generados por la aplicación Swing Evolution en cuanto a su apariencia visual. | 20 | 5 | 2 | 0 | 27 |
| En lo que concierne al tiempo de diseño para la personalización de los componentes Swing como lo calificaría. | 15 | 11 | 1 | 0 | 27 |
| Qué tal le parecen los nuevos componentes Swing generados por la aplicación. | 13 | 13 | 1 | 0 | 27 |
| Cree que esta aplicación es una buena herramienta de trabajo para el programador, para la creación de interfaces gráficas con una nueva apariencia visual. | 27 | 0 | 0 | 0 | 27 |
| **RENDIMIENTO** | | | | | |
| En cuanto al rendimiento cómo calificaría a la aplicación. | 21 | 5 | 1 | 0 | 27 |
| En la creación de los componentes visuales como cataloga usted a la aplicación Swing Evolution. | 24 | 2 | 1 | 0 | 27 |

| | | | | | |
|---|---|---|---|---|---|
| Cuál sería su calificación en lo referente a la creación y generación de código del componente personalizado en la aplicación. | 22 | 3 | 2 | 0 | 27 |
| La ejecución de la aplicación Swing Evolution en la plataforma Windows como usted la calificaría. | 18 | 2 | 2 | 5 | 27 |
| La ejecución de la aplicación Swing Evolution en la plataforma Linux como usted la calificaría. | 23 | 4 | 0 | 0 | 27 |
| En la ejecución de las tareas de limpieza, compilación y generación de módulos .JAR del proyecto cuál es su calificación. | 20 | 1 | 6 | 0 | 27 |
| **Resultado** | **256** | **73** | **17** | **5** | **351** |

## GUÍA QUE SE UTILIZÓ PARA REALIZAR LAS PRUEBAS A LOS USUARIOS DEL SISTEMA

Para saber si la aplicación cumple con las perspectivas propuestas al inicio del proyecto se realizaron pruebas de validación y de rendimiento las cuales se las aplicó a los docentes y estudiantes, quienes siguieron unos procedimientos o pasos para llegar al resultado de la aplicación que es de crear nuevos componentes swing y acoplarlos a la plataforma NetBeans.

**A continuación se presenta los procesamientos o pasos:**

- Se crea un nuevo proyecto, eligiendo en el menú “Archivo” ítem “Nuevo Proyecto”, se elige el componente que se va a personalizar, los nombres del proyecto y del componente.

- Si el componente es un Button o un ToogleButton se le puede cambiar la forma del mismo, la cual se la logra de la siguiente manera:

    - Se visualiza la “Paleta Formas” eligiéndola en el menú “Ventanas” ítem “Paleta Formas”.
    - Se arrastra la figura deseada desde la “Paleta Formas” hacia el diseñador cuantas veces lo desee.
    - Para posicionarse en las diferentes figuras que se arrastró se escoge en la “Paleta de Combinaciones” los botones “Activación de Figura” y en la figura elegida se la puede mover, agrandarla o achicarla, moverla al centro del diseñador.
    - Para combinar 2 figuras se elige en la “Paleta de Combinaciones” se elige en numero de las figuras a combinar, la operación lógica a aplicarse y se elige la opción “Probar Operación”, en este caso todavía es para manipular las dos figuras al gusto, y cuando estén listas las figuras elegir la opción “Combinar”.

- Para agregar los efectos se los aplica de la siguiente manera:

    - Se visualiza la “Paleta Efectos” eligiéndola en el menú “Ventanas” ítem “Paleta Efectos”.
    - Se elige el tipo de degradado (horizontal, vertical, diagonal derecha, diagonal izquierda, diagonal derecha, efecto 3D), en el cual se presenta el dialogo para elegir los colores del degradado.
    - Se elige los colores deseados y se los acepta.

- Después de estar conforme se procede a compilar el proyecto haciendo click derecho sobre el proyecto en la “Paleta Proyectos” y se elige “Clean & Build”, paralelamente presenta un mensaje de que se ha creado el .jar del proyecto en la “Paleta Ejecución”

- A continuación se agrega el módulo generado del componente (.jar) a la plataforma NetBeans para poder utilizarlo en el mismo en creación de interfaces gráficas de usuario (Ventanas).

**6.5.1.- ANALISIS DE PRUEBAS.-**

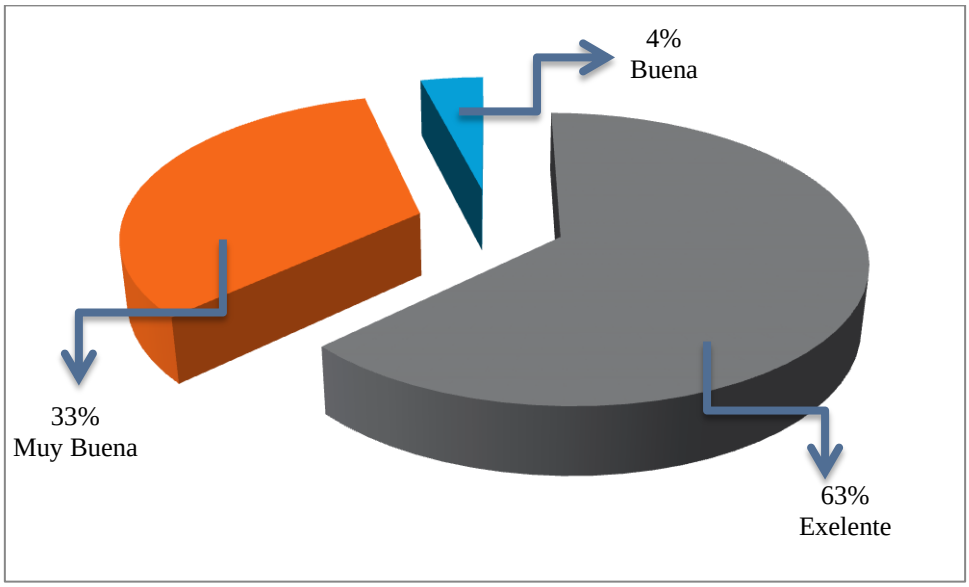
Una vez termina las pruebas realizadas a los docentes y estudiantes se obtienen los siguientes resultados:

**Swing Evolution 1.1.0.**

**1. ¿Cuál sería su calificación acerca del Ambiente de Desarrollo Swing Evolution en lo que corresponde a su Entorno Gráfico?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelentes. | 17 | 63 |
| 2 | Muy Buena. | 9 | 33 |
| 3 | Buena. | 1 | 4 |
| 4 | Regular. | 0 | 0 |

| | | |
|---|---|---|
| **TOTAL** | **27** | **100** |

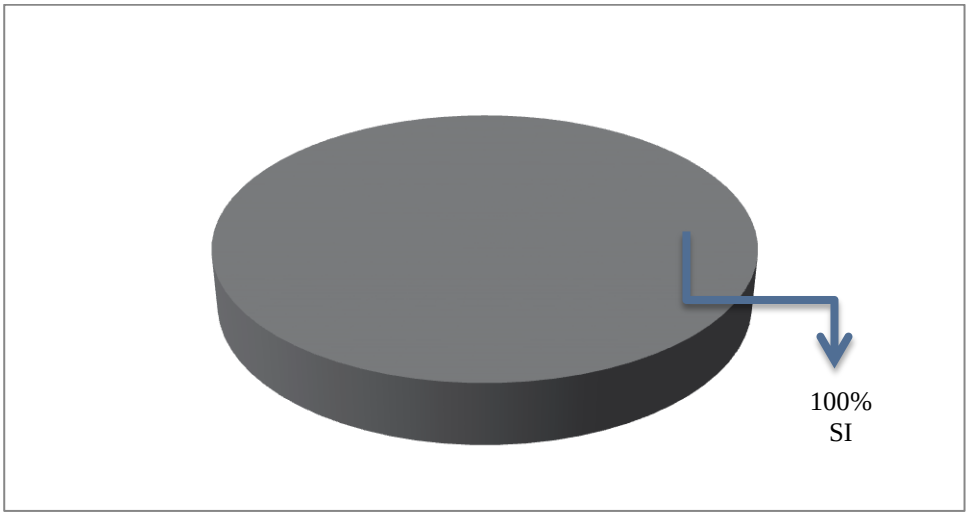



❖ En lo que respecta a la distribución de las diferentes paletas que tiene Swing Evolution se puede evidenciar en el cuadro y gráfico estadístico que predomina la excelencia con un 63%, Muy buena con un 33,0%, buena con un 4% y Regular con un 0%.

❖ Esto nos quiere decir que Swing Evolution tiene una excelente distribución de paletas en su entorno y por lo tanto son fáciles de encontrarlas para trabajar a gusto.

**2. ¿La aplicación te parece amigable al usuario?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| **1** | **Si.** | **27** | **100** |
| **2** | **No.** | **0** | **0** |
| | **TOTAL** | **27** | **100** |



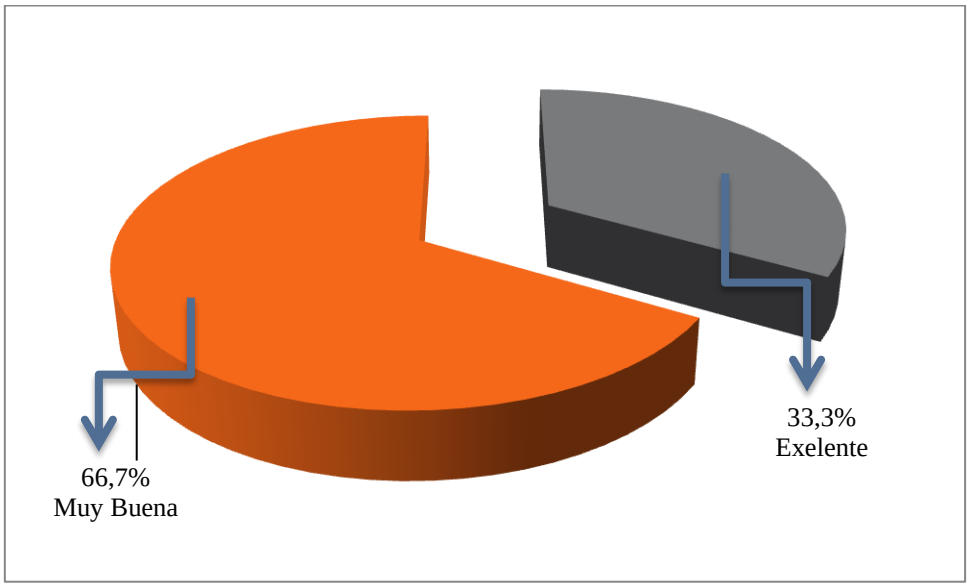

❖ Como se puede evidenciar en el cuadro y gráfico estadístico predomina que Swing Evolution es amigable al usuario con un 100%.

❖ Entonces podemos afirmar que Swing Evolution es amigable al usuario en el sentido que se pueden encontrar fácilmente las paletas de trabajo ya que están agrupadas según su función en un lugar específico de la aplicación, se ejecutan pocos pasos para personalizar los componentes, presenta información de la generación de los componentes, etc.

**3. ¿Cómo le parece la distribución de los diferentes controles visuales dentro de la aplicación Swing Evolution?**

| N$^{ro}$ | OPCIONES | F | % |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **1** | **Excelentes.** | **9** | **33,3** |
| **2** | **Muy Buena.** | **18** | **66.7** |
| **3** | **Buena.** | **0** | **0** |
| **4** | **Regular.** | **0** | **0** |
| | **TOTAL** | **27** | **100** |

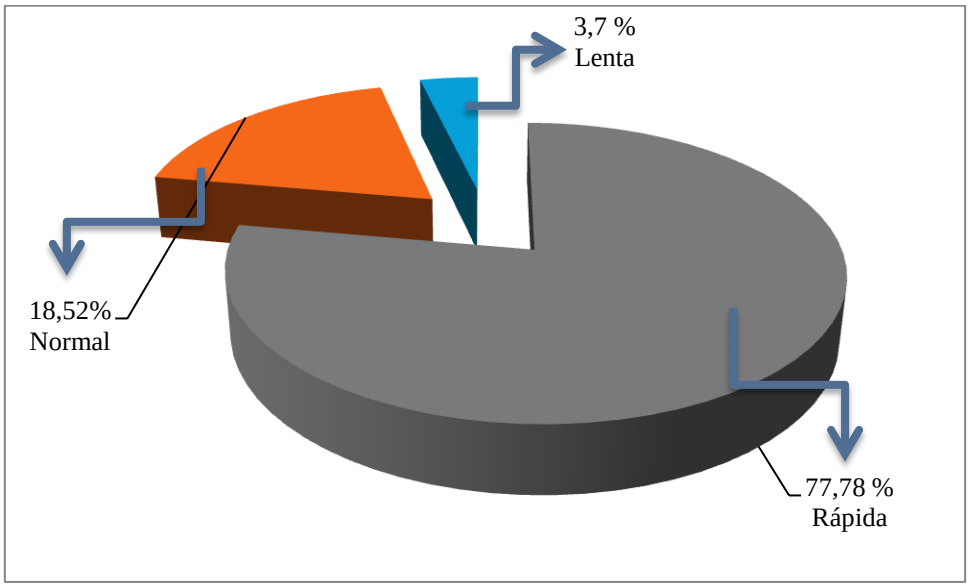



- Como se puede evidenciar en el cuadro y gráfico estadístico tiene muy buena distribución de los controles visuales dentro de su entorno con un 66,70 % de los encuestados que lo afirman, y un 33,30% de los encuestados que dicen que la distribución es excelente.

- En este caso la distribución de los controles dentro del entorno de Swing Evolution es muy buena ya que las paletas de trabajo están agrupadas según su función y son fáciles de encontrar y utilizar.

**4. ¿En cuanto a rendimiento cómo calificaría a la aplicación?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Rápida. | 21 | 77,78 |
| 2 | Normal. | 5 | 18,52 |
| 3 | Lenta. | 1 | 3,70 |
| | TOTAL | 27 | 100 |

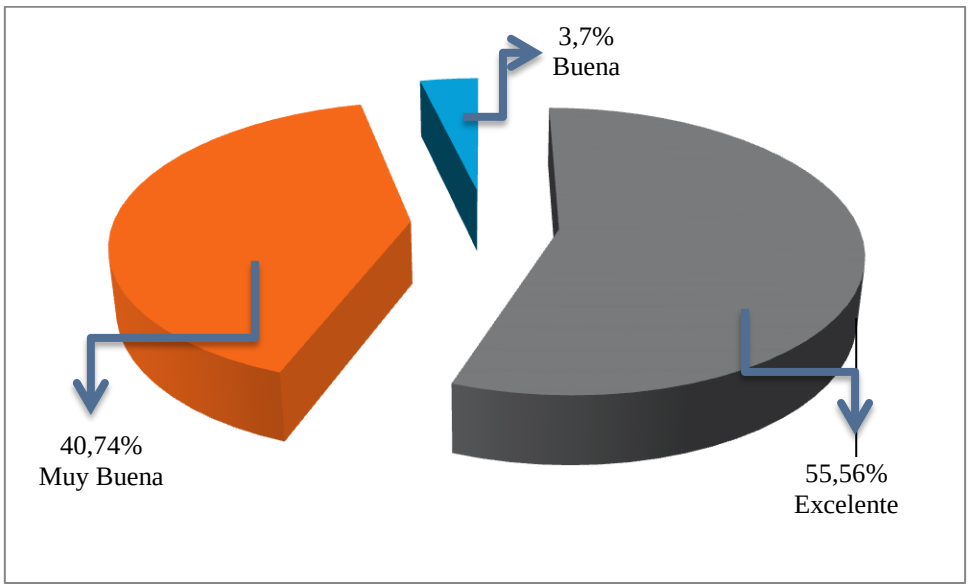



- Como se puede observar en el cuadro y gráfico estadístico existe una mayoría de usuarios que lo califican a Swing Evolution como rápida con un 77,78% en sus procesos para crear nuevos componentes, un 18,52% de los encuestados opinan el rendimiento es normal y 3,7% Lenta.

- Con los porcentajes expuestos se puede afirmar que Swing Evolution tiene un rendimiento aceptable para la creación de nuevos componentes Swing ya sea en los procesos de generación de código, compilación de fuentes, generación del binario .jar, etc.

**5. ¿En lo que concierne al tiempo de diseño para la personalización de los componentes Swing como lo calificaría?**

| Nro | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelente. | 15 | 55,56 |
| 2 | Muy Buena. | 11 | 40,74 |
| 3 | Buena. | 1 | 3,70 |
| 4 | Regular. | 0 | 0 |
| | TOTAL | 27 | 100 |

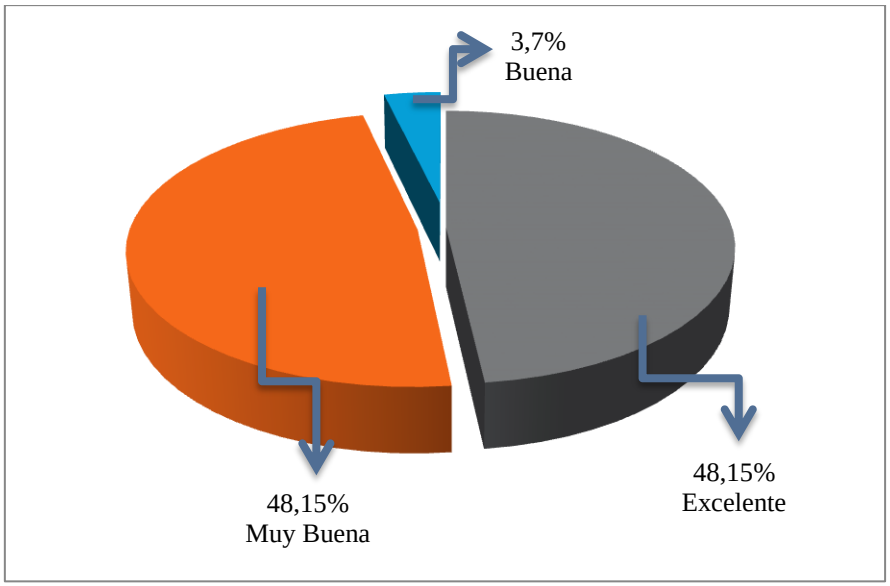


- Como se puede observar en el cuadro y gráfico estadístico existe un 55,56% de los encuestados que afirman que Swing Evolution tiene un excelente tiempo de creación de nuevos componentes swing, como también un 40,74% nos dice que el tiempo es muy bueno y un 3,70% dice que el tiempo de creación es bueno.

- Con los datos estadísticos presentados podemos decir que Swing Evolution mejora el tiempo de programación de los componentes Swing personalizados a nuestro gusto.

**6. ¿Qué tal le parecen los nuevos componentes Swing generados por la aplicación?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelentes. | 13 | 48,15 |
| 2 | Muy Buena. | 13 | 48,15 |
| 3 | Buena. | 1 | 3,70 |
| 4 | Regular. | 0 | 0 |
| | TOTAL | 27 | 100 |
| | | | |

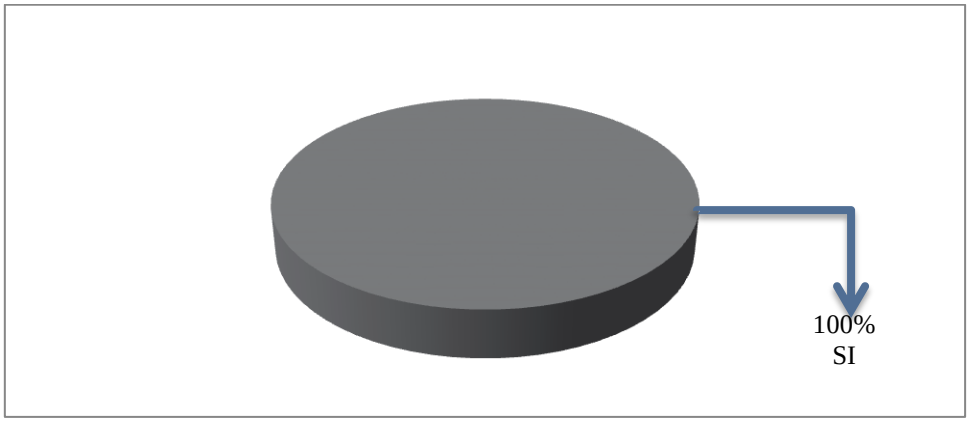



- Como se puede apreciar en el cuadro y gráfico estadístico un 48,15% de los encuestados nos dice que los componentes generados por Swing Evolution son excelentes, un 48,15% nos dice que son muy buenos y un 3,70 % nos dice que son buenos.

- Con los datos estadísticos presentados podemos decir que los componentes creados por Swing Evolution son excelentes y que se los puede utilizar en la creación de interfaces gráficas de usuario y dar una vista agradable al usuario.

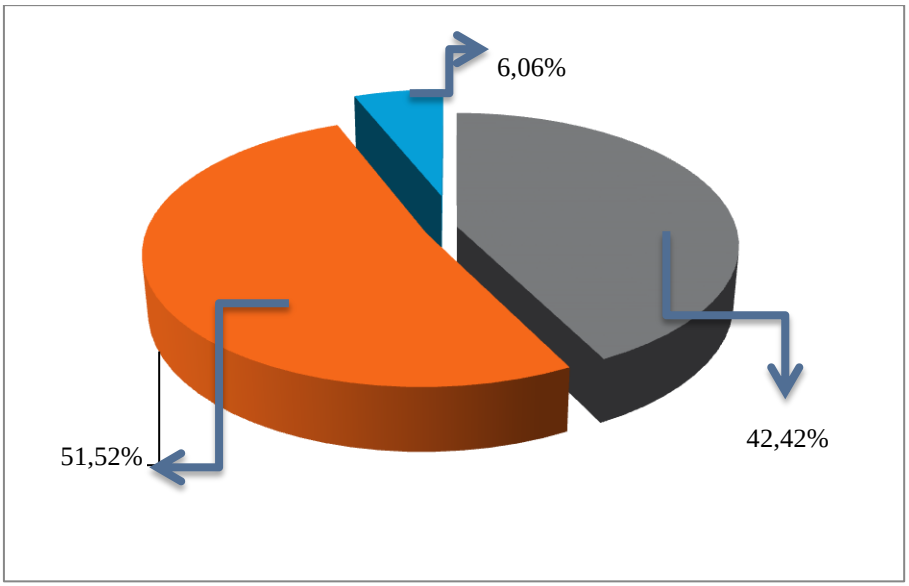
7. **¿Cree que esta aplicación es una buena herramienta de trabajo para el programador, para la creación de interfaces gráficas con una nueva apariencia visual?**

| Nro | OPCIONES | f | % |
|---|---|---|---|
| 1 | Si. | 27 | 100 |
| 2 | No. | 0 | 0 |
| | TOTAL | 27 | 100 |




**Porque:**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Ahorra el tiempo de desarrollo. | 14 | 42,42 |
| 2 | Explora capacidades Java2D. | 17 | 51,52 |
| 3 | Mejora la presentación de las aplicaciones. | 2 | 6,06 |
| | TOTAL | 33 | 100 |



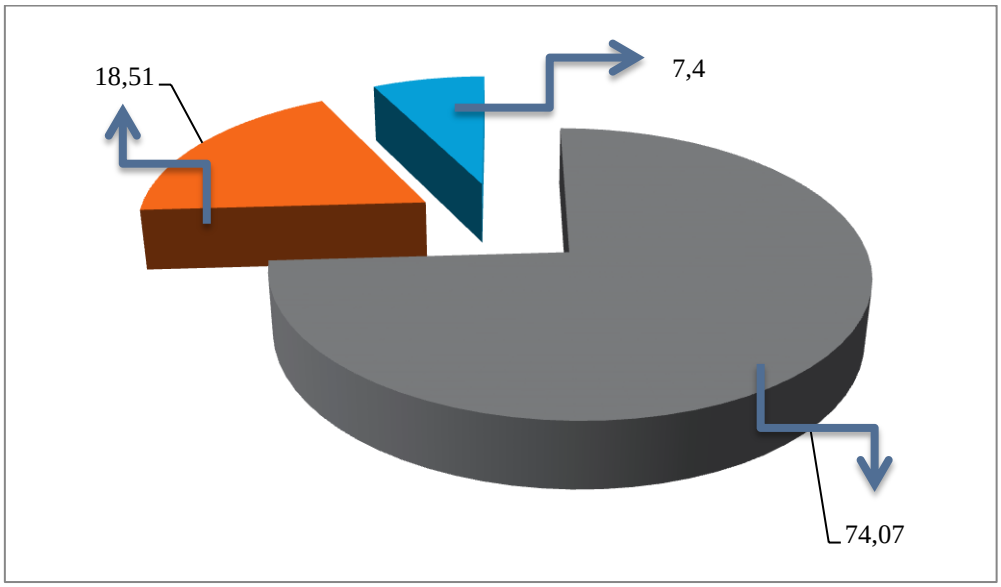

- Observando el cuadro y gráfico estadístico se puede decir que la totalidad de los encuestados creen que Swing Evolution es una buena herramienta de trabajo para la creación de interfaces gráficas con una nueva apariencia visual.

- Presentados los datos estadísticos se puede afirmar que Swing Evolution es una muy buena herramienta para la creación de interfaces gráficas con una nueva apariencia visual ya que permite entre otras cosas:

    - Ahorrar el tiempo de desarrollo.
    - Explorar capacidades 2D.
    - Mejorar la presentación de las aplicaciones

8. **¿Cómo calificaría a cada uno de los componentes generados por la aplicación Swing Evolution en cuanto a su apariencia visual?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelente. | 20 | 74,07 |
| 2 | Muy Buena. | 5 | 18,51 |
| 3 | Buena. | 2 | 7,40 |
| 4 | Regular. | 0 | 0 |
| | TOTAL | 27 | 100 |

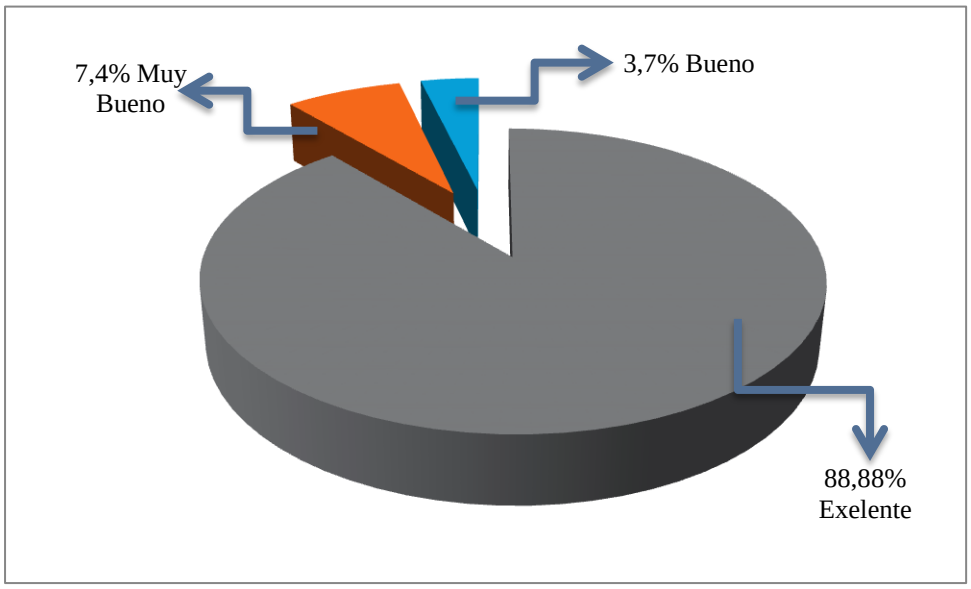



- Como se puede observar en el cuadro y gráfico estadístico un 74,07% de los encuestados dicen que la apariencia de los componentes creados por Swing Evolution son excelentes, un 18,51% dicen que son buenos y un 7,4% dicen que son buenos.

- Se puede afirmar que la apariencia de los mayoría de los componentes creados por Swing Evolution son excelentes

9. **¿En la creación de los componentes visuales como cataloga usted a la aplicación Swing Evolution?**

| Nro | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelentes. | 24 | 88,88 |
| 2 | Muy Buena. | 2 | 7,40 |
| 3 | Buena. | 1 | 3,70 |
| 4 | Regular. | 0 | 0 |
| | TOTAL | 27 | 100 |

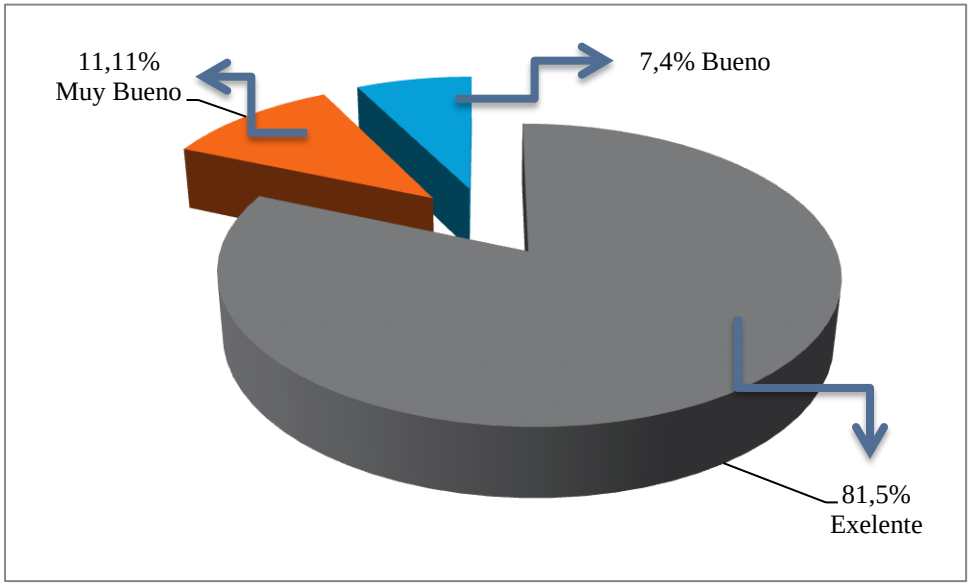



- En el cuadro y gráfico estadístico se puede observar que un 88,8% de los encuestados opinan que la manera de creación de nuevos compones Swing por parte de swing Evolution es excelente, un 7,4% opina que son muy buenos y un 3,7% son buenos.

- Con la presentación de los resultados afirmamos que la manera de creación de los nuevos componentes Swing por parte de Swing Evolution es exelente ya que genera componentes ligeros al memento de acoplarlos a la plataforma NetBeans.

**10. ¿Cuál sería su calificación en lo referente a la creación y generación de código del componente personalizado en la aplicación?**

| Nro | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelentes. | 22 | 81,5 |
| 2 | Muy Buena. | 3 | 11,11 |
| 3 | Buena. | 2 | 7,40 |
| 4 | Regular. | 0 | 0 |
| | TOTAL | 27 | 100 |

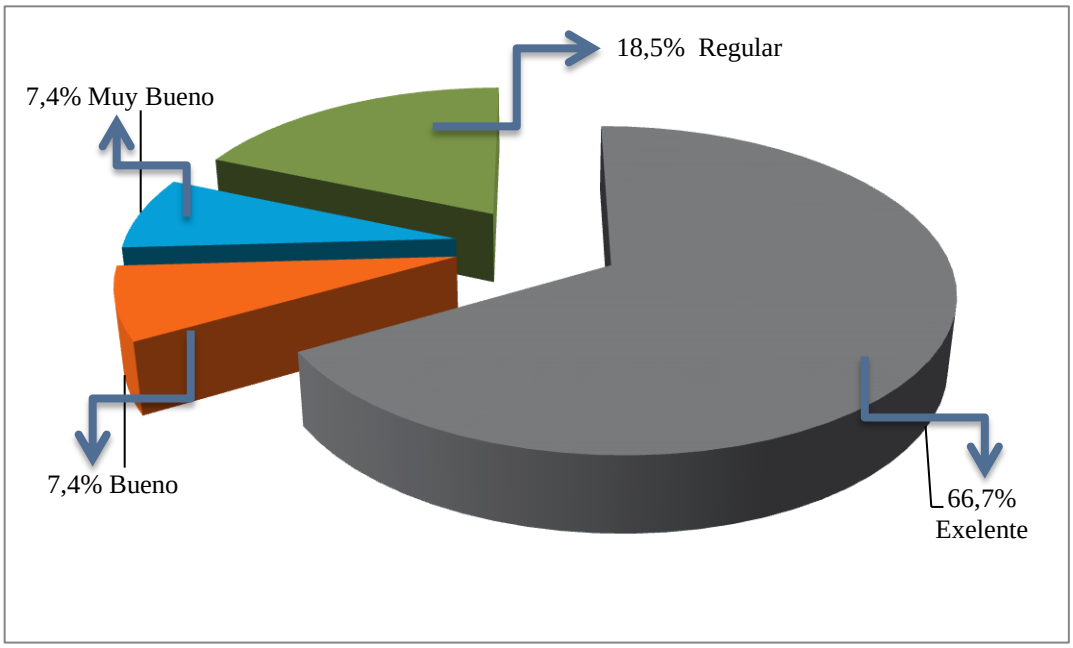


- Como se puede observar en el cuadro y gráfico estadístico un 81,5% de los encuestados dicen que la generación de código fuente de los componentes creados en Swing Evolution es exelente, un 11,11% de los encuestados dicen que es muy bueno y un 7,4% de los encuestados nos dicen que es bueno.

- Presentados los datos estadísticos podemos decir que la generación de código fuente de los componentes creados a través de Swing Evolution es excelente ya que es comprensible al usuario.

**11. ¿La ejecución de la aplicación Swing Evolution en la plataforma Windows como usted la calificaría?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelente. | 18 | 66,7 |
| 2 | Muy Bueno. | 2 | 7,40 |
| 3 | Bueno. | 2 | 7,40 |
| 4 | Regular. | 5 | 18,5 |
| | TOTAL | 27 | 100 |



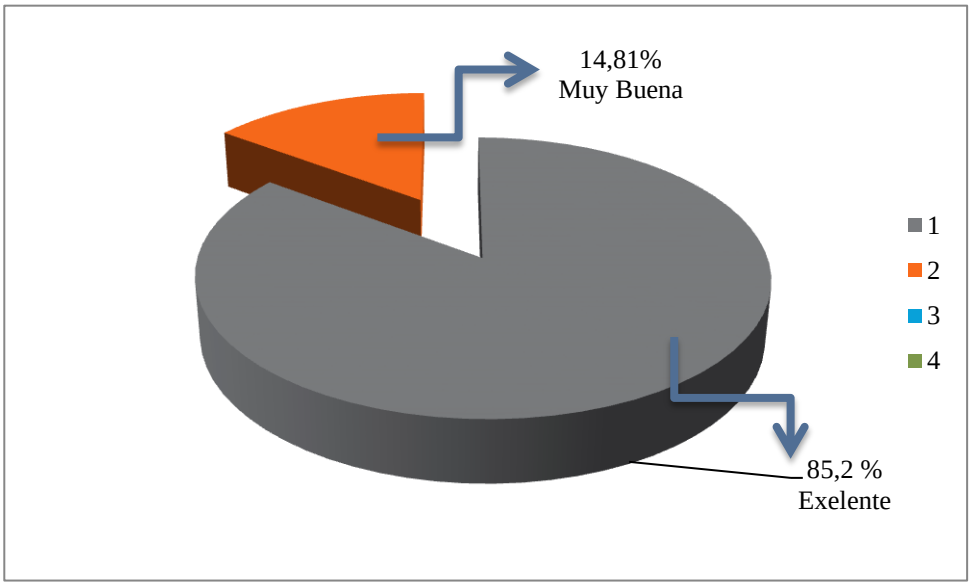

- Como se puede apreciar en el cuadro y gráfico estadístico un 66,7% de los encuestados opinan que la ejecución de Swing Evolution en la Plataforma Windows es excelente, un 18,5% de los encuestados opinan que es regular, un 7,4% de los encuestados opinan que es muy bueno, y un 7,4% de los encuestados opinan que es bueno.

- Se puede evidenciar que la ejecución de Swing Evolution en la plataforma Windows es muy aceptable ya que el tiempo de ejecución con respecto a otros entornos similares es inferior.

**12. ¿La ejecución de la aplicación Swing Evolution en la plataforma Linux como usted la calificaría?**

| Nro | OPCIONES | f | % |
|---|---|---|---|
| 1 | Excelentes. | 23 | 85,2 |
| 2 | Muy Buena. | 4 | 14,81 |
| 3 | Buena. | 0 | 0 |

| | | | |
|---|---|---|---|
| **4** | **Regular.** | **0** | **0** |
| | **TOTAL** | **27** | **100** |



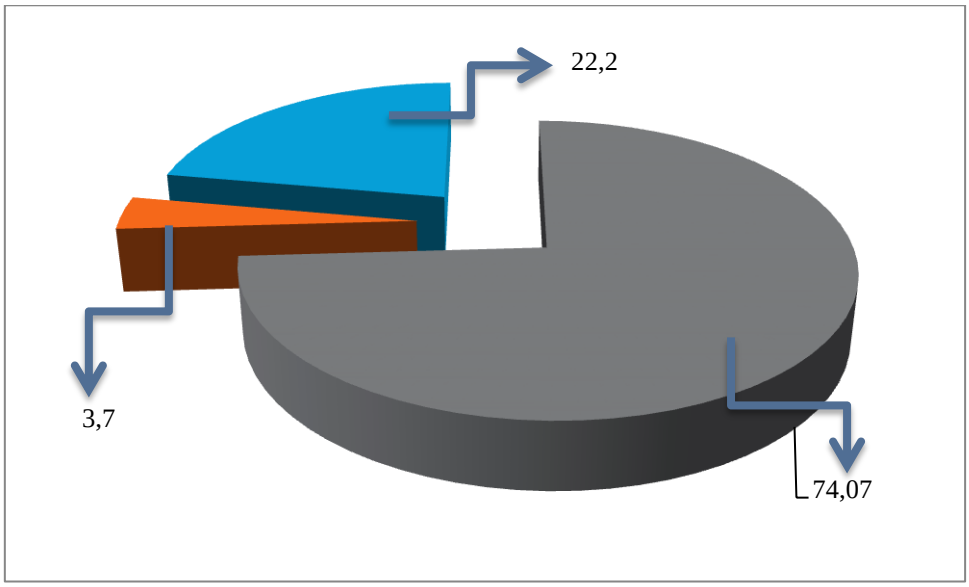

- Como se puede apreciar en el cuadro y gráfico estadístico un 85,2% de los encuestados opina que la ejecución de Swing Evolution en la plataforma Linux es excelente y un 14,81% de los encuestados opinan que es muy bueno.

- Presentados los datos estadísticos se puede afirmar que la ejecución de Swing Evolution en la plataforma Linux es excelente y que trabaja de manera normal.

**13. ¿En la ejecución de las tareas de limpieza, compilación y generación de módulos .JAR del proyecto cuál es su calificación?**

| N$^{ro}$ | OPCIONES | f | % |
|---|---|---|---|
| **1** | **Excelentes.** | **20** | **74,07** |

| | | | |
|---|---|---|---|
| **2** | **Muy Buena.** | **1** | **3,70** |
| **3** | **Buena.** | **6** | **22,22** |
| **4** | **Regular.** | **0** | **0** |
| | **TOTAL** | **27** | **100** |




- Como se puede observar en el cuadro y gráfico estadístico un 74,07% de los encuestados dicen que las tareas para generación de los módulos .jar son excelentes, un 22,22% de los encuestados nos dicen que son buenas y un 3,70 de los encuestados nos dicen que son muy buenas.

- Como se puede apreciar la ejecución de tareas que realiza Swing Evolution para trabajar con los módulos .jar es excelente ya que la generación de .jar lo hace de forma automatizada.

## 7. VALORACIÓN TÉCNICA ECONÓMICA.-

En la culminación del presente proyecto se ha evidenciado que la aplicación cumple con los objetivos planteados al inicio del mismo, entre las características que posee la aplicación están:

- Mejora el tiempo de desarrollo en la creación de componentes swing.
- Mejora de manera positiva la apariencia visual de las interfaces gráficas de usuario.
- La aplicación es amigable al usuario final, programador.
- Genera componentes adaptables a las plataformas Netbeans.
- Gran aceptación por los estudiantes de la carrera.

Así mismo en lo que concierne a la inversión económica del presente proyecto se ha evidenciado que los costos son mínimos, debido a que se ha tomado en cuenta lo que conocemos actualmente como software Libre GNU.

A continuación se detalla los recursos utilizados para la elaboración del proyecto:

### 7.1.- RECURSOS.-

- **HUMANOS.-** A continuación detallamos los recursos humanos que intervienen directa e indirectamente en el proyecto:

**RECURSOS HUMANOS**

| Descripción | Cantidad | # Horas | Valor Unitario | Costo Total |
|---|---|---|---|---|
| Director de Tesis | **1** | **0** | **0.00** | **0.00** |

| | | | | |
|---|---|---|---|---|
| Programadores Analistas | **2** | **500** | **0.80** | **800** |
| | | | **Total =** | **$ 800.00** |

| **Descripción** | **Cantidad** | **# Horas** | **Valor Unitario** | **Costo Total** |
|---|---|---|---|---|
| **HARDWARE** | | | | |
| Computador Portátil DELL Vostro 1500 | **1** | | | **1700** |
| Computador Portátil DELL Latitude D531 | **1** | | | **1500** |
| Impresora | **1** | **30** | **60.00** | **60.00** |
| Escáner | **1** | **10** | **45.00** | **45.00** |
| Dispositivos USB | **2** | **500** | **30.00** | **60.00** |
| **SOFTWARE** | | | | |
| Netbeans 5.5 | **1** | **1000** | **0.00** | **0.00** |
| Plataforma Java ( **J2SDK**) | **1** | **1000** | **0.00** | **0.00** |
| Microsoft Word | **1** | **100** | **0.00** | **0.00** |
| Enterprise UML | **1** | **320** | **0.00** | **0.00** |

| | | | | |
|---|---|---|---|---|
| | | | **Total =** | **$ 3365** |

- **TÉCNICOS.-** Los recursos técnicos que se utilizarán en el desarrollo de nuestro proyecto se describe a continuación:

- **MATERIALES.-** A continuación se describe los recursos materiales que se utilizarán para el desarrollo del proyecto.

**RECURSOS MATERIALES**

| Descripción | Cantidad | # Horas | Valor Unitario | Costo Total |
|---|---|---|---|---|
| Cartuchos de Tinta | **10** | **0** | **2.50** | **25** |
| Transporte | | | **100** | **100** |
| Resma de Papel | **4** | **0** | **5.00** | **20.00** |
| Copias | **500** | **0** | **0.03** | **15.00** |
| Internet | **1** | **500** | **0.80** | **400.00** |
| Anillados | **5** | **0** | **1.00** | **5.00** |
| Libros | **2** | **0** | **20.00** | **40.00** |
| | | | **Total =** | **605.00** |

- **ECONOMICOS.-**

  Los Recursos Económicos para la elaboración del Proyecto son los siguientes:

  - ✓ **Recursos Humanos** **$ 800.00**
  - ✓ **Recursos Técnicos** **$ 3365.00**
  - ✓ **Recursos Materiales** **$ 605.00**

  **Total $ 4770.00**

- **TECNOLÓGICOS.-**

Los recursos tecnológicos que se utilizarán para el desarrollo del proyecto serán despreciables con respecto al beneficio que se obtendrá, debido a que estas tecnologías son en algunos de los casos gratuitas.

- ✓ Tecnología JAVA (**J2SDK 1.6.0**).-
- ✓ Editores Desarrollo Netbeans 5.5 Lenguajes Java.
- ✓ Tecnologías Java 2D y 3D.
- ✓ JDom
- ✓ Ant-Apache

## 8. CONCLUSIONES.-

Al término del presente proyecto se concluye que:

- En la actualidad no existen herramientas de desarrollo que ayuden a los programadores a mejorar la apariencia visual de las controles Swing.

- La aplicación Swing Evolution permite explorar las potencialidades que nos ofrece la tecnología Java2D.

- La apariencia 3D de los componentes generados por la aplicación se la puede simular a través de la tecnología Java2D.

- El código generado (Archivos **.java**) de los componentes creados por la aplicación Swing Evolution es comprensible y legible para el programador.

- Se pueden integrar los componentes creados por Swing Evolution a otros IDEs basados en la tecnología JAVABEANS, como es el caso del Entorno de Desarrollo Integrado Netbeans.

- EL Framework Apache-Ant es una de las herramientas que nos permite la automatización de procesos tales como: limpieza, compilación, generación y ejecución de módulos (JAR).

- La aplicación Swing Evolution tiene un buen rendimiento en los dos sistemas operativos que se ha comprobado cómo son Linux y Windows.

- Realizadas las encuestas a los docentes y alumnos de la Carrera de Ingeniería en Sistemas se ha evidenciado que la aplicación tiene una excelente acogida entre los mismos.

## 9. RECOMENDACIONES.-

Finalizado el presente proyecto de tesis se recomienda que:

- Los estudiantes de la Carrera de Ingeniería en Sistemas hagan uso de la aplicación, como una herramienta de apoyo para la creación de Interfaces Graficas de Usuario (GUI) a través de la utilización de herramientas como la plataforma Netbeans.

- Para la ejecución de la aplicación Swing Evolution se pide tener los siguientes requisitos mínimos de Hardware y software:

    - **Procesador Intel Pentium IV 1.6 GHZ**
    - **Memoria Ram: 512MB.**
    - **Disco Duro: 1GB Espacio libre disponible.**
    - **J2SE versión 1.5.0 o posteriores.**
    - **Netbeans 5.0 y posteriores.**

- La difusión de la herramienta Swing Evolution se la haga mediante la licencia **GPL**, para su mejoramiento y futuras versiones de esta aplicación.

- Se estudie código fuente generado por la aplicación para investigar más a fondo las cualidades de las tecnologías utilizadas.

  Que los docentes de la Carrera de Ingeniería en Sistemas fomenten en sus aulas la utilización y difusión de esta herramienta.

- Que la aplicación no sea solamente difundida a nivel local, sino también que se la haga a través del Internet en la página oficial de la Universidad Nacional de Loja; con la finalidad de recibir comentarios y sugerencias para su mejoramiento.

## 10. BIBLIOGRAFIA.-


**Libros.-**

- **CHET HAASE Y ROMAIN GUY, 2008-10-12, Filthy Rich Clients. Primera Edición, New York, Editorial: Addison-Wesley, 250 páginas.**

**Sitios Web.-**

- http://www.adictosaltrabajo.com
- http://www.comunity.java.net
- http://www.comunidadjava.com.ar
- http://www.elhacker.com
- http://www.filthrichclients.org
- http://www.forosweb.com
- http://www.google.com
- http://www.java.net
- http://www.java2s.com
- http://www.lawebdelprogramador.com
- http://www.netbeans.org
- http://www.opensource.jdk.com
- http://www.ozito.com
- http://www.programacioncastellano.com
- http://www.sun.com
- http://www.toptutoriales.com
- http://www.wikipedia.org

# 11. ANEXOS

# ANEXO 1: ANTEPROYECTO

## 1.1.- TEMA.-

“Diseño e Implementación de una Herramienta Visual que permita la creación de nuevos Componentes SWING utilizando Java 2D Y 3D”

## 1.2.- SITUACIÓN PROBLEMÁTICA.-

### 1.2.1.- ANTECEDENTES.-

La tecnología Java se creó como una herramienta de programación en una pequeña operación secreta y anónima denominada "The Green Project" en Sun Microsystems en el año 1991.

El equipo secreto ("Green Team"), compuesto por trece personas y dirigido por James Gosling, se encerró en una oficina desconocida de Sand Hill Road en Menlo Park, interrumpió todas las comunicaciones regulares con Sun y trabajó sin descanso durante 18 meses.

Intentaban anticiparse y prepararse para el futuro de la informática. Su conclusión inicial fue que al menos en parte se tendería hacia la convergencia de los dispositivos digitales y los ordenadores. Pero poco tiempo después Internet estaba listo para la tecnología Java y, justo a tiempo para su presentación en público en 1995, el equipo pudo anunciar que el navegador Netscape Navigator incorporaría la tecnología Java.

Actualmente, a punto de cumplir los 10 años de existencia, la plataforma Java ha atraído a cerca de 4 millones de desarrolladores de software, se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, ordenadores y redes de cualquier tecnología de programación.

Swing es una biblioteca gráfica para Java que forma parte de las *Java Foundation Classes* (JFC). Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

Las *Internet Foundation Classes* (IFC) eran una biblioteca gráfica para el lenguaje de programación Java desarrollada originalmente por Netscape y que se publicó en 1996.

Desde sus inicios el entorno Java ya contaba con una biblioteca de componentes gráficos conocida como AWT. Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. Entonces una aplicación Java corriendo en Windows usaría el botón estándar de Windows y una aplicación corriendo en UNIX usaría el botón estándar de Motif. En la práctica esta tecnología no funcionó:

- Al depender fuertemente de los componentes nativos del sistema operativo el programador AWT estaba confinado a un mínimo denominador común entre ellos. Es decir que solo se disponen en AWT de las funcionalidades presentes en todos los sistemas operativos.

- El comportamiento de los controles varía mucho de sistema a sistema y se vuelve muy difícil construir aplicaciones portables. Fue por esto que el eslogan de Java "Escríbalo una vez, ejecútelo en todos lados" fue parodiado como "Escríbalo una vez, pruébelo en todos lados".

En cambio, los componentes de JFC eran mostrados y controlados directamente por código Java independiente de la plataforma. De dichos

componentes se dice con frecuencia que son componentes ligeros, dado que no requieren reservar recursos nativos del sistema de ventanas del sistema operativo. Además al estar enteramente desarrollado en Java aumenta su portabilidad asegurando un comportamiento idéntico en diferentes plataformas.

En 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras tecnologías de las Java Foundation Classes. Además de los componentes ligeros suministrados originalmente por la IFC, Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma.

Swing es un conjunto de paquetes construido en la parte más alta de AWT que proporciona un gran número de clases pre construidas, aproximadamente 250 clases y 40 componentes. Desde punto de vista del programador, los componentes UI son probablemente los más interesantes, obsérvese que cada componente UI de Swing empieza con J, que es una de las razones por la que muchos programadores usan erróneamente los términos JFC y Swing intercambiándolos:

- Japplet: Versión extendida de java.applet que añade soporte para los paneles base y otros paneles.
- JButton: Pulsador o comando de botón, etc.

Swing es un extenso conjunto de componentes que van desde los más simples, como etiquetas, hasta los más complejos, como tablas, árboles, y documentos de texto con estilo. Casi todos los componentes Swing descienden de un mismo padre llamado JComponent que desciende de la clase de AWT Container. Es por ello que Swing es más una capa encima de AWT que una sustitución del mismo. Si la compara con la jerarquía de Component notará que para cada componente AWT hay otro equivalente en Swing que empieza con

"J". La única excepción es la clase de AWT Canvas, que se puede reemplazar con JComponent, JLabel, o JPanel.

La característica más notable de los componentes Swing es que están escritos al 100% en Java y no dependen de componentes nativos, como sucede con casi todos los componentes AWT. Esto significa que un botón Swing y un área de texto se verán y funcionarán idénticamente en las plataformas Macintosh, Solaris, Linux y Windows. Este diseño elimina la necesidad de comprobar y depurar las aplicaciones en cada plataforma destino.

### 1.2.2.- PROBLEMÁTICA.-

En la actualidad, los problemas más comunes que los programadores tienen a la hora de desarrollar software en la plataforma Java están vinculados directamente a la Vista, GUI (Interfaz Gráfica de Usuario). Es así que en el mundo del software los componentes Swing de la plataforma Java ayudan de alguna manera a los programadores a mejorar la creación de Interfaz Gráfica de Usuario para así poder tener un producto amigable acorde a las exigencias del usuario.

Los problemas más relevantes que radican en este tipo de componentes JFC/SWING es la pérdida de tiempo a la hora de su personalización ya que los desarrolladores de la plataforma Java deben acoplarse a la apariencia normal-básica del J2SDK (Máquina Virtual de Java), es por esta razón que el tiempo de desarrollo para la personalización de cada uno de los componentes Swing se hace cada vez más difícil. Así mismo los programadores tienen que escribir muchas líneas de código para poder lograr obtener una personalización eficaz de los componentes.

Un problema que también se presentan muy común en el proceso de la personalización de los componentes SWING es la escritura del código fuente, el mismo que en algunos casos no puede ser entendido de manera rápida por los programadores lo cual les ocasiona confusión en el proceso de aprendizaje.

En la actualidad las herramientas que están manejando los programadores de la plataforma Java se han encargado de la personalización de la GUI (Interfaz Gráfica de Usuario) de manera superficial y básica ,es por ello que vemos la necesidad del Desarrollo de una herramienta de Diseño Gráfico de componentes Swing basados en la tecnología Java2D y Java3D.

## 1.3 Problema de investigación:

***"Falta de Herramientas Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D"***

### 1.3.1.- Delimitación del problema:

A causa que ahora en la actualidad no existen herramientas que permitan a los desarrolladores y programadores modificar la apariencia predeterminada de los componentes Swing hemos creído conveniente realizar este proyecto como es la creación de un Generador de nuevos componentes Swing Java, en el cual consta de las siguientes etapas de Desarrollo : Análisis, Diseño, Implementación y Pruebas, para que de esta manera se pueda cumplir con el objetivo tanto General como los objetivos específicos planteados en el proyecto.

Dicho proyecto constará con una planificación adecuada por cada una de las etapas antes mencionadas, en cada una ellas se irán evaluando los puntos más críticos y así poder obtener un proyecto que satisfaga las necesidades de los desarrolladores y programadores.

El presente proyecto será realizado a través de herramientas de programación de Software Libre como es la plataforma JAVA, a través de la tecnología J2SE, para modelado del proyecto se utilizará herramientas UML como es Enterprise Architect o Poseidón 2.1.

Este proyecto también tiene como finalidad ser una aplicación orientada al aprendizaje por parte de los señores estudiantes del Área de Energía, Industrias y Recursos Naturales no Renovables, mas específicamente en la Carrera de Ingeniería en Sistemas, con el afán de colaborar de alguna manera con el proceso de enseñanza-aprendizaje en nuestra carrera.

## 1.4.- Justificación

La Universidad Nacional de Loja, es una institución educativa de gran reconocimiento a nivel nacional, la misma que cuenta con cinco áreas orientadas a la capacitación de bachilleres, cuyo objetivo es preparar profesionales con una gran capacidad crítica que les permita aportar alternativas de solución en beneficio de la sociedad.

Entre las áreas que conforman esta institución, destacaremos el Área de Energía, Industrias y Recursos Naturales no Renovables cuyas actividades

encaminan al crecimiento, enriquecimiento y desarrollo de entes sociales que reciben una educación científica, técnica y tecnológica en diferentes carreras tales como: Ing. en Sistemas, Geología Ambiental y Ordenamiento Territorial, Electricidad, Electrónica, Construcción y Mecánica Automotriz.

La carrera de Ingeniería en Sistemas forma profesionales con conocimientos profundos de la estructura y particularidades del software, y para llevar a la práctica todos estos conocimientos adquiridos durante los 6 años de carrera universitaria hemos decidido realizar un proyecto que permita a los programadores en la plataforma Java desarrollar de manera rápida nuevos componentes personalizados basados en la forma básica predeterminada por Java del paquete **javax.swing** a través de un Wizard de Desarrollo de componentes y de esta manera dar una nueva y moderna apariencia personalizada a todos los componentes que este paquete java contiene.

### 1.4.1 Justificación Académica

La Universidad Nacional de Loja a través del SAMOT, propicia el logro de aprendizajes significativos, en un proceso de vinculación permanente de la docencia, la investigación y la extensión, en base de las problemáticas y demandas de la sociedad.

Es así que, tomando como pilar fundamental la extensión nos orientamos a vincular la teoría con la práctica, aplicando todos los conocimientos adquiridos durante nuestra vida universitaria, es así que hemos decidido crear un Generador de nuevos componentes Swing Java con apariencia 2D y 3D. La finalidad primordial mediante la elaboración de nuestro proyecto es obtener el respectivo título de Ingeniero en Sistemas.

### 1.4.2 Justificación Técnica

Ahora en la actualidad el avance de la tecnología ha ido evolucionando a pasos muy extensos como es el mundo del Software, muy particularmente en los avances tecnológicos que han surgido en la plataforma Java. La vinculación de la tecnología Java a los proyectos de Desarrollo de Software en la actualidad se han constituido una de las pilares fundamentales para la elaboración de nuevas tecnologías que pretenden dar a los desarrolladores, programadores y usuarios finales nuevas herramientas que puedan ayudar a mejorar el tiempo de desarrollo de cualquier tipo de proyecto de software.

La vinculación de esta tecnología Java a nuestro proyecto es muy importante ya que con el uso de la misma se podrá ayudar a los programadores a mejorar el tiempo de desarrollo en la creación de aplicaciones y así también obtener una mejoramiento en la parte gráfica visual de los componentes.

### 1.4.3 Justificación Operativa

Es factible realizar este proyecto porque ayudará a los programadores y desarrolladores de la plataforma Java a tener una variedad de componentes Swing para la creación de sus propias aplicaciones y de la misma manera poder satisfacer las necesidades visuales de los usuarios de las aplicaciones.

### 1.4.4 Justificación Económica.

Un paso importante antes de iniciar nuestro proyecto es realizar un estudio de factibilidad económica con el cual sustentaremos la importancia que tiene para los programadores y desarrolladores en la plataforma java la creación de esta herramienta de desarrollo.

Para el presente proyecto nosotros como desarrolladores contamos con los suficientes recursos tanto económicos, humanos y bibliográficos que permitan finalizar con éxito el proyecto planteado.

## 1.5.- OBJETIVOS DE LA INVESTIGACIÓN.-

### 1.5.1.- OBJETIVO GENERAL.-

- Crear de una Herramienta Visual para la personalización de Componentes JFC/SWING

### 1.5.2.- OBJETIVOS ESPECÍFICOS:

- Mejorar el Tiempo de Desarrollo en el modelamiento de los componentes de las aplicaciones Swing con tecnologías Java 2D y 3D

- Crear una Herramienta de fácil uso al programador para la creación de nuevos Componentes Swing basados en los componentes básicos.

- Permitir la creación de componentes visuales Swing personalizados para el programador.

- Generar el Código fuente en Java de los componentes visuales.

- Lograr el acoplamiento de los nuevos componentes Swing generados por nuestra aplicación en el entorno de desarrollo integrado Netbeans 5.0 y versiones posteriores.

- Implementar la herramienta en la carrera de Ingeniería en Sistemas del Área de Energía, las Industrias y los Recursos Naturales no Renovables.

## *2.0.- MARCO TEÓRICO.-*

### *2.1.- Introducción.-*

Java 2D, al igual que todo el lenguaje Java permite la portabilidad de programas lo que le permite a los desarrolladores crear programas que son independientes de la plataforma en la cual se les ejecutará. Esta característica de Java 2D le da a las aplicaciones implementadas en este lenguaje una flexibilidad que de otro modo no tendrían. Además esta facilidad se ha llevado en Java 2D al nivel de que también es independiente de sobre qué tipo de dispositivo gráfico se está trabajando.

Java 2d extiende las capacidades contenidas en el paquete de AWT (Abstract Windowin Tool) de Java. Sin embargo se mantiene compatible con este heredar las características de aquel. Esto permite que programas escritos para AWT puedan ejecutarse con Java 2D.

Java 2D se mantiene fiel a la lógica imperante en el ambiente Java, por consiguiente permite a los desarrolladores manipular los objetos creados bajo Java 2D con la misma seguridad que todos los objetos de Java

### *2.2.- Objetivos*

Los Objetivos que se proponían satisfacer los desarrolladores de Java 2D correspondieron a mejorar la calidad gráfica de las aplicaciones programadas en Java. Anteriormente existía el ambiente AWT pero no estaba al mismo nivel que los demás ambientes de programación, en cuanto a eficiencia se refiere.

El siguiente objetivo correspondió a satisfacer la necesidad para los programadores de permitirles tener una sola interfaz de programación independiente de si se estuviese imprimiendo en una impresora o un monitor. Es decir Java2D desde un comienzo buscaba hacer transparente para el programador el dispositivo final sobre el que se estaba trabajando para permitirle enfocarse tan sólo en las características propiamente gráficas del programa en desarrollo.

Por último los desarrolladores se planteaban el objetivo de mantener una heterogeneidad entre las distintas componentes de Java de modo que para los programadores sea intuitivo el aprender a utilizar las nuevas herramientas en desarrollo. Esto implica el mantener la compatibilidad con las herramientas más antiguas como AWT.

### *2.3.- Características.-*

Las características del manejo que hace Java2D de objetos gráficos son bastante avanzados, dentro de ellos se distinguen tres categorías: el manejo de gráficos, de texto y de imágenes.

En cuanto al manejo de gráficos, Java 2D trae clases para manipularlos de manera avanzada. Entre otras destaca la posibilidad de aplicar antialising y ocupar distintos tipos de trazos, además se pueden manejar transparencias y sobre posiciones de los gráficos. También se permiten modelar objetos gráficos complejos por medio de líneas de Bezier. Hay disponible una gran cantidad de operaciones que se pueden efectuar sobre los gráficos que incluyen rotaciones, deformaciones, escalamiento y traslaciones.

Para el manejo de texto se incluyen herramientas que permiten la manipulación de junturas de letras, necesidad relevante para representar determinados idiomas. Además de que se define un espacio para escribir en la pantalla. Las fuentes a las que accede Java2D corresponden a las fuentes del sistema sobre el cual esta implementada la aplicación, para lo cual también se incorporan una serie de herramientas.

La característica más avanzada de Java 2D la representa el manejo de imágenes. Para mejorar la eficiencia en este sentido, Java2D incorporó un nuevo modelo de despliegue de la imagen, el nuevo paradigma se caracteriza por mantener la imagen en memoria y hacer las operaciones sobre este objeto, de este modo se mejora el rendimiento en cuanto al tiempo de ejecución de las operaciones. Además de esta característica, Java2D incorpora una gran gama de filtros que se pueden aplicar de manera sencilla sobre las imágenes para poder lograr los efectos deseados por el implementador.

## 2.4.- JAVA 3D.-

El Java 3D API es una interfase para escribir programas que desplieguen gráficas tridimensionales además de interactuar con ellas. Java 3D es una extensión del Java 2 JDK. El API contiene un conjunto de constructores de alto nivel para crear y manipular geometrías en 3D, así como de las técnicas necesarias para dibujar la geometría en una imagen. Java 3D contiene las funciones necesarias para crear imágenes, visualizaciones, animaciones y aplicaciones con objetos gráficos interactivos.

### *2.4.1.- ¿Que es Java 3D?*

El Java 3D API es un conjunto de clases que sirven como interface para dibujar gráficas sofisticadas en tercera dimensión utilizando sonido, si se desea.

Java 3D es un lenguaje para programar aplicaciones que contengan gráficos en tres dimensiones perfectamente interactivos, una herramienta de alto nivel para crear y manipular geometría en 3D, así como imágenes y animaciones con objetos gráficos. Este libro presenta un panorama completo para aprender a programar en 3D con Java 3D. Desde la instalación del programa, las jerarquías de las clases, pasando por la creación de animaciones elementales, construcción de geometrías, hasta el empleo de efectos y sonidos. Un libro que describe paso a paso cada una de las funciones de Java 3D.

Describe las relaciones de usuario: crear y utilizar una clase Behavior, condiciones de activación, navegación por teclado, activación y navegación por ratón.

Explica los diferentes tipos de iluminación: la total, la restrictiva y el empleo de las sombras; también presenta la aplicación de texturas: texels, cargador de texturas, filtros, atributos, corrección de perspectiva, hasta multitexturas

### *2.4.2.- El Java 3D API.-*

Todo programa en Java 3D es un conjunto de objetos ordenados en base a una serie de jerarquías basada en la herencia. Esta colección de objetos describe a un espacio virtual que es la guía para dibujarlos en una imagen posteriormente. Este API contiene aproximadamente 100 clases contenidas dentro del paquete *javax.media.j3d*. A estas clases por lo general se les conoce como el núcleo (core) de Java 3D.

Aunque existen cientos de variables y métodos dentro de las clases de Java 3D, en el universo virtual se pueden crear animaciones utilizando unas pocas clases, que es lo que se enseñará en esta sección.

Además de las clases de Java 3D se utilizarán las clases del paquete *com.sun.j3d.utils* conocidas como las utilerías de Java 3D, ya que en estas se encuentran las clases de más bajo nivel, por lo que son necesarias para cualquier programa en Java 3D.

Estas utilerías tienen cuatro categorías: contenedores, constructores de escena, clases de geometrías y utilerías. En un futuro se incorporarán NURBS. Estas utilerías reducen significativamente el número de líneas de código, ya que además se hace uso de las clases contenidas en el *java.awt* y *javax.vecmath.* El paquete *java.awt* define el Abstarct Windowing Toolkit (AWT) que sirve para crear la ventana en donde se despliega la imagen. El paquete *javax.vecmath* define una clase matemática para realizar operaciones con puntos, vectores, matrices y otras funciones matemáticas.

# 3.-DISEÑO METODOLÓGICO.-

## 3.1 Tipo de estudio.-

Para la elaboración y recolección de información para nuestra investigación utilizaremos la metodología hipotética deductiva, el cual consiste que a partir de una Hipótesis la cual será fundamentada por la práctica y la experiencia de los involucrados en el proyecto.

El método deductivo se basa fundamentalmente en el seguimiento de pasos sistemáticos y sencillos en la investigación, la cual ayudarán a los investigadores a tener una perspectiva clara y concisa de las necesidades que posee el objeto de la investigación.

Así mismo el método inductivo tiene una gran ventaja para la recolección de la información acerca de los elementos de la investigación, de cuanta información se dispone de dichos objetos, el tipo de información que se tratará de discrepar para tomar las mejores resoluciones y contrastar con el objeto de la investigación.

Para el desarrollo de nuestro proyecto también utilizaremos una metodología de Entrega por Etapas del Ciclo de Vida del Desarrollo del Software, la misma que consiste en mostrar el proyecto de software a los clientes en etapas refinadas sucesivamente, lo que diferencia este método de los muchos que existen es que cuando se utiliza el modelo de Entrega por Etapas, se conoce exactamente qué es lo que se va construir cuando se procede a construirlo.

Con el método de Entrega por Etapas se atraviesan los pasos del modelo convencional pasando por la definición del concepto del software, análisis de requerimientos y creación del diseño global de una Arquitectura para el programa completo que se intenta construir. Luego de esto se procede a realizar el diseño detallado, la codificación, depuración y pruebas dentro de cada una de las etapas.

En lo concerniente a la experiencia investigativa y científica se tratara de realizar la fundamentación teórica y práctica de lo que se pretende realizar a través de la investigación.

### 3.2.- Población, muestra y Unidades de Observación.-

Para la recolección de información de nuestro proyecto la población esta constituida por los programadores, desarrolladores, diseñadores gráficos, docentes y estudiantes en informática, los cuales se encuentran vinculados directamente en el mundo del Desarrollo de Software a través de la plataforma Java.

En cuanto a la muestra de la población serán los anteriormente mencionados los que nos proporcionarán la información suficiente y necesaria para la realización del cálculo estadístico de la muestra poblacional.

### 3.3.- Métodos e Instrumentos. –

Las técnicas o métodos para la recopilación de la información tenemos los siguientes:

- Investigación Bibliográfica.
- Encuestas.
- Observación.

La investigación bibliográfica tendrá como finalidad obtener la mayor información posible y actualizada para la creación de nuestro proyecto, a fin de cumplir con las necesidades y problemas más relevantes que durante el desarrollo del proyecto surjan.

Las encuestas serán dirigidas a los docentes y estudiantes de la Carrera de Ingeniería en Sistemas de AEIRNNR, para de esta manera obtener una visión general de la problemática que se pretende resolver.

En lo concerniente a la observación será aplicada en las unidades relacionadas con la programación Interfaces Gráficas, etc. que los docentes y estudiantes realizan durante las etapas de Desarrollo de Proyectos de Software, para de esta manera poder investigar a fondo los problemas que estas presentan.

### 3.4.- Plan de tabulación y análisis de la información.

Se debe planificar previamente una serie de cuadros estadísticos, matrices para procesar la información o la elaboración de software que posibiliten el resumir la información que se recogerá con la aplicación de los instrumentos seleccionados.

### 3.5 Elaboración o redacción del informe y de las alternativas de solución

Se presenta un esquema general para la presentación del documento técnico producto del proceso de investigación.

## 4. ORGANIZACIÓN Y GESTIÓN DE LA INVESTIGACIÓN

En el proceso de Investigación y recolección de información para la elaboración de nuestro proyecto de Desarrollo, se ha tenido encuesta los siguientes factores muy importantes que incurren directamente en el desarrollo sistemático y objetivo del fin que se desea alcanzar.

Es así que se ha considerado de la manera más importante los siguientes parámetros que se describen a continuación:

### 4.1 RECURSOS.-

**4.1.1 HUMANOS.-** A continuación detallamos los recursos humanos que intervienen directa e indirectamente en el proyecto:

**RECURSOS HUMANOS**

| **Descripción** | **Cantidad** | **# Horas** | **Valor Unitario** | **Costo Total** |
|---|---|---|---|---|
| Director de Tesis | **1** | **0** | **0.00** | **0.00** |
| Investigadores | **2** | **3** | **0.80** | **144.00** |
| Asesor | **2** | **20** | **5.00** | **200.00** |
| Diseñadores Gráficos | **1** | **10** | **10** | **100.00** |
| Programadores Analistas | **2** | **640** | **0.00** | **0.00** |
| | | | **Total =** | **444.00** |

**4.1.2 TÉCNICOS.-** Los recursos técnicos que se utilizarán en el desarrollo de nuestro proyecto se describe a continuación:

| **Descripción** | **Cantidad** | **# Horas** | **Valor Unitario** | **Costo Total** |
|---|---|---|---|---|
| **HARDWARE** | | | | |
| Computador | **1** | **640** | **850** | **850.00** |
| Impresora | **1** | **30** | **60.00** | **60.00** |
| Escáner | **1** | **10** | **45.00** | **45.00** |
| Dispositivos USB | **2** | **500** | **30.00** | **60.00** |
| **SOFTWARE** | | | | |
| Netbeans 5.5 | **1** | **1000** | **0.00** | **0.00** |
| Plataforma Java ( **J2SDK**) | **1** | **1000** | **0.00** | **0.00** |
| Microsoft Word | **1** | **100** | **0.00** | **0.00** |
| Enterprise UML | **1** | **320** | **0.00** | **0.00** |
| | | | **Total =** | **1015.00** |

**4.1.3 MATERIALES.-** A continuación se describe los recursos materiales que se utilizarán para el desarrollo del proyecto.

**RECURSOS MATERIALES**

| Descripción | Cantidad | # Horas | Valor Unitario | Costo Total |
|---|---|---|---|---|
| Cartuchos de Tinta | **10** | **0** | **2.50** | **25** |
| Transporte | | | **100** | **100** |
| Resma de Papel | **4** | **0** | **5.00** | **20.00** |
| Copias | **500** | **0** | **0.03** | **15.00** |
| Internet | **1** | **500** | **0.80** | **400.00** |
| Anillados | **5** | **0** | **1.00** | **5.00** |
| Libros | **2** | **0** | **20.00** | **40.00** |
| | | | **Total =** | **605.00** |

**4.1.4 ECONOMICOS.-**

Los Recursos Económicos para la elaboración del Proyecto son los siguientes:

- ✓ **Recursos Humanos** **$ 444.00**
- ✓ **Recursos Técnicos** **$ 1015.00**
- ✓ **Recursos Materiales** **$ 605.00**

| | | |
|---|---|---|
| **Total** | **$** | **2064.00** |

### 4.1.5 **TECNOLÓGICOS.-**

Los recursos tecnológicos que se utilizarán para el desarrollo del proyecto serán despreciables con respecto al beneficio que se obtendrá, debido a que estas tecnologías son en algunos de los casos gratuitas.

- ✓ Tecnología JAVA (**J2SDK 1.6.0**).-
- ✓ Editores Desarrollo Netbeans 5.5 Lenguajes Java.
- ✓ Tecnologías Java 2D y 3D.
- ✓ JDom
- ✓ Ant-Apache

ANEXO A1: MATRICES

# MATRICES PARA EL DISEÑO DEL PROYECTO

## MATRIZ DE CONSISTENCIA GENERAL

**ENUNCIADO DE LA PROBLEMÁTICA:** ***"Falta de Herramientas Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D"***

| TEMA | PROBLEMA GENERAL | OBJETO DE INVESTIGACION | OBJETIVO GENERAL | OBJETIVOS ESPECÍFICOS | HIPOTESIS |
|---|---|---|---|---|---|
| Diseño e Implementación de | Falta de Herramientas | Programación en Java 2D y Java 3D | Diseño e Implementación | **1.-** Mejorar el Tiempo de Desarrollo en el | Con la personalización |

| | | | | | |
|---|---|---|---|---|---|
| un Wizard para la Personalización de Componentes Swing 2D Y 3D para el Lenguaje de Programación JAVA<br><br>Diseño e Implementación de un Wizard para la Personalización de | Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D | para la personalización de Componentes Swing. | de una Herramienta Visual para la personalización de Componentes JCF/SWING<br><br>Diseño e Implementación de una Herramienta Visual para la | modelamiento de los componentes de las aplicaciones Swing con Tecnologías Java 2D y 3D.<br><br>**2.-** Crear una Herramienta de fácil uso al programador para la creación de nuevos Componentes Swing basados en los componentes básicos. | de estos componentes las aplicaciones serán más atractivas y amigables a los usuarios finales mejorando la<br><br>acogida del software desarrollado en la plataforma Java. |

| | | | | | |
|---|---|---|---|---|---|
| Componentes Swing 2D Y 3D para el Lenguaje de Programación JAVA | Falta de Herramientas Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D | Programación en Java 2D y Java 3D para la personalización de Componentes Swing | personalización de Componentes JCF/SWING<br><br>Diseño e Implementación | **3.-** Permitir la creación de los componentes visuales Swing personalizados para el programador.<br><br>**4.-** Generar código fuente. En Java de los componentes visuales.<br><br>**5.-** Lograr el acoplamiento y utilización de los nuevos | |

| | | | | | |
|---|---|---|---|---|---|
| Diseño e Implementación de un Wizard para la Personalización de Componentes Swing 2D Y 3D para el Lenguaje de Programación JAVA | Falta de | | de una Herramienta Visual para la personalización de Componentes JCF/SWING<br><br>Diseño e Implementación de una Herramienta | componentes Swing generados por nuestra aplicación en el Entorno de Desarrollo Integrado Netbeans 5.0 y versiones posteriores<br><br>**6.-** Implementar la Herramienta en la Carrera de Ingeniería en Sistemas del Area de Energía, | |

| | | | | | |
|---|---|---|---|---|---|
| Diseño e Implementación de un Wizard para la Personalización de Componentes Swing 2D Y 3D para el Lenguaje de Programación JAVA | Herramientas Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D | Programación en Java 2D y Java 3D para la personalización de Componentes Swing | Visual para la personalización de Componentes JCF/SWING | las Industrias y los Recursos Naturales no Renovables. | |

| | | | | | |
|---|---|---|---|---|---|
| | Falta de Herramientas Visuales para la personalización de Componentes Swing Java con apariencia 2D y 3D | Programación en Java 2D y Java 3D para la personalización de Componentes Swing | | | |

| | | | | | |
|---|---|---|---|---|---|

**MATRIZ DE CONSISTENCIA ESPECÍFICA**

| OBJETIVO ESPECÍFICO | PROBLEMA ESPECÍFICO | ALTERNATIVA DE SOLUCIÓN | SISTEMA CATEGORIAL |
|---|---|---|---|
| Mejorar el Tiempo de Desarrollo en el modelamiento de los componentes de las aplicaciones Swing con Tecnologías Java 2D y 3D. | Para la personalización de cualquier componente Swing se requiere de mucho tiempo y esfuerzo por parte del programador, ya que tiene que codificar mucho para si lograr la apariencia deseada. | Diseño e Implementación de un Wizard para la Personalización del árbol de Componentes SWING | • Introducción a Java 2D y 3D.<br>• Importancia de tecnologías Java 2D y 3D<br>• Tipos de componentes JFC/Swing<br>• Software para tecnología 3D<br>• Aplicaciones desarrolladas en 2D y 3D.<br>• Software de Desarrollo (IDE Programación) |
| Crear una Herramienta | No existen | | |

| | | | |
|---|---|---|---|
| de fácil uso al programador para la creación de nuevos<br><br>Componentes Swing basados en los componentes básicos.<br>. | herramientas de Software en el mercado que<br><br>permitan a los<br><br>programadores en Java a lograr de manera rápida y eficiente la personalización de los Componentes Swing | Diseño e<br><br>Implementación de un<br><br>Wizard para la<br><br>Personalización del<br><br>árbol de Componentes<br><br>SWING | • API de j2sdk 1.6.0 y API de Java 2D y 3D<br>• Software de Aplicación KIT J2SDK 1.6.0 y Java 3D<br>• Introducción a Swing.<br>• Información del Árbol de Componentes Swing.<br><br>• IDE de Programación en Java: Netbeans 5.5<br>  • Estructura de las Clases de los componentes.<br>  • Herramientas de Modelamiento de Clases.<br>  • Software de Modelamiento de Clases. |
| Permitir la creación de los componentes visuales Swing personalizados para el programador. | El programador debe acoplarse a la apariencia básica por defecto de la plataforma java. | Crear una herramienta visual para poder manipular la apariencia normal de los componentes Swing de | • Nociones de Diseño Gráfico.<br>• Editores de Diseño como Photoshop 8.0<br>• Formas Básicas (Plantillas) de los componentes.<br>• API de j2sdk 1.6.0 y API de Java 2D y 3D<br>• Software de Aplicación KIT J2SDK 1.6.0 y Java 3D<br>• Introducción a Swing. |

| | | | |
|---|---|---|---|
| | | manera rápida mejorando el tiempo de desarrollo. | • Información del Árbol de Componentes Swing. |
| Generar código fuente. En Java de los componentes visuales.<br><br>Lograr el acoplamiento y utilización de los nuevos componentes Swing generados por nuestra aplicación en el Entorno de Desarrollo Integrado Netbeans 5.0 y versiones posteriores | El código generado por algunos IDE de Programación es poco legible para su lectura y comprensión<br><br>Existen algunos IDE de programación que no soportan la agregación de nuevos componentes. | Crear un herramienta visual que genere código fuente lo más legible posible para los programadores<br><br>Utilizar los IDE de programación que soporten la agregación de nuevos componentes Swing. | • Introducción a Java 2D y 3D.<br>• Importancia de tecnologías Java 2D y 3D<br>• Tipos de componentes JCF/Swing<br>• Software para tecnología 3D<br>• Software de Desarrollo (IDE Programación)<br><br>• IDE de Programación Netbeans.<br>• Compilación en Caliente |

| | | | |
|---|---|---|---|
| | | | |
| Implementar la Herramienta en la Carrera de Ingeniería en Sistemas del Area de Energía, las Industrias y los Recursos Naturales no Renovables | Ninguno | Implementación de la Herramienta. | |

**MATRIZ DE OPERATIVIDAD**

**OBJETIVO ESPECÍFICO:**

***"Mejorar el Tiempo de Desarrollo en el modelamiento de los componentes de las aplicaciones Swing con tecnologías Java 2D y 3D"***

| Actividad o tarea | Metodología | Fecha | | Responsables | Presupuesto | Resultados esperados |
|---|---|---|---|---|---|---|
| Búsqueda bibliográfica de las Tecnologías Java 2D y 3D | Se consultará en Bibliotecas e Internet, manuales, libros, tutoriales relacionados con las Tecnologías Java 2D y 3D. | 02/08/07 | 06/08/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 40 | Información detallada de las Tecnologías Java 2D y 3D. |
| Análisis para la creación del Código fuente generado por el IDE de | Analizar, documentar y desarrollar algoritmos que permitan a nuestro IDE de programación generar código fuente legible y comprensible para el programador | 09/08/07 | 13/08/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 60 | Obtener algoritmos especializados para la generación de |

| | | | | | | |
|---|---|---|---|---|---|---|
| Programación. | | | | | | código fuente |
| Codificación y pruebas de Algoritmos desarrollados | Codificación de los algoritmos para la generación de código fuente. | 16/08/07 | 20/08/07 | Egdo. Darwin Calle.<br><br>Egdo. Daniel Quezada | $ 40 | Obtener de Algoritmos funcionales para nuestro IDE en la generación de C. |

**OBJETIVO ESPECÍFICO:**

***<u>" Crear una Herramienta de fácil uso al programador para la creación de nuevos Componentes Swing basados en los componentes básicos "</u>***

| **Actividad o tarea** | **Metodología** | **Fecha** | | **Responsables** | **Presupuesto** | **Resultados esperados** |
|---|---|---|---|---|---|---|
| Búsqueda bibliográfica del árbol de Componentes Swing. | Se consultará en Bibliotecas e Internet, manuales, libros, tutoriales relacionados con el paquete Swing. | 23/08/07 | 25/08/07 | Egdo. Darwin Calle.<br><br>Egdo. Daniel Quezada | $ 40 | Información detallada de los Componentes Swing. |

| | | | | | | |
|---|---|---|---|---|---|---|
| Análisis del Sistema | Recopilar información acerca del tema planteado y realizar un análisis profundo de los objetivos que se desea alcanzar. | 26/08/07 | 18/09/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 60 | Obtener una ideas claras de cómo el sistema va ha cumplir los objetivos planteados. |
| Diseño de Clases. | Analizar, documentar y construir nuestro de Diagrama de Clases.<br>Realizar el Análisis del Dominio. | 21/09/07 | 08/10/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 40 | Obtener el Diagrama Clases, comprensible para la etapa P. |

| | | | | | | |
|---|---|---|---|---|---|---|
| Implementación | Codificación de nuestro diagrama de Clases | 11/10/07 | 28/03/08 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 300 | Obtener un prototipo del sistema casi funcional. |
| | Realizar las respectivas pruebas del sistema atreves de la | 01/04/08 | 30/04/08 | Egdo. Darwin Calle. | $ 100 | Obtener un sistema |

| | | | | | | |
|---|---|---|---|---|---|---|
| Pruebas | utilización de la herramienta por diversos programadores. | | | Egdo. Daniel Quezada | | funcional en su totalidad con sus respectivos cambios realizados. |

OBJETIVO ESPECÍFICO:

***“ Generar Código fuente Java de los componentes Visuales ”***

| Actividad o tarea | Metodología | Fecha | | Responsables | Presupuesto | Resultados esperados |
|---|---|---|---|---|---|---|
| Búsqueda bibliográfica de las Tecnologías Swing Java 2D y 3D | Se consultará en Bibliotecas e Internet, manuales, libros, tutoriales relacionados con las Tecnologías Java 2D y 3D. | 02/08/07 | 06/08/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 40 | Información detallada de las Tecnologías Java 2D y 3D. |
| Análisis para la creación del Código fuente generado por el IDE de Programación. | Analizar, documentar y desarrollar algoritmos que permitan a nuestro IDE de programación generar código fuente legible y comprensible para el programador | 09/08/07 | 13/08/07 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 60 | Obtener algoritmos especializados para la generación de código fuente |

| | | | | | | |
|---|---|---|---|---|---|---|
| Codificación y pruebas de Algoritmos desarrollados | Codificación de los algoritmos para la generación de código fuente. | 16/08/07 | 20/08/07 | Egdo. Darwin Calle.<br><br>Egdo. Daniel Quezada | $ 40 | Obtención de Algoritmos funcionales para nuestro IDE en l |

OBJETIVO ESPECÍFICO:

***“ Lograr el acoplamiento y utilización de los nuevos componentes Swing generados por nuestra aplicación en el Entorno de Desarrollo Integrado Netbeans 5.0 y versiones posteriores “***

| **Actividad o tarea** | **Metodología** | **Fecha** | | **Responsables** | **Presupuesto** | **Resultados esperados** |
|---|---|---|---|---|---|---|
| Búsqueda bibliográfica del código fuente del IDE de programación Netbeans 5.0 y versiones posteriores. | Se consultará en Internet, manuales, tutoriales relacionados con el IDE de programación Netbeans. | 24/02/08 | 27/02/08 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 40 | Información de cómo agregar nuevos componentes al IDE de programación Netbeans. |
| Análisis del Código fuente del IDE de Programación | Analizar, documentar y acoplar el código generado por nuestra aplicación para que cumpla con las especificaciones que requiere la plataforma Netbeans. | 01/04/08 | 11/04/08 | Egdo. Darwin Calle.<br>Egdo. Daniel Quezada | $ 60 | Lograr el acoplamiento satisfactorio de nuestros componentes al IDE de |

| | | | | | | |
|---|---|---|---|---|---|---|
| Netbeans. | | | | | | programación Netbeans. |
| OBJETIVO ESPECÍFICO:<br><br>***<u>“ Implementar la Herramienta en la Carrera de Ingeniería en Sistemas del Área de Energía, las Industrias y los Recursos Naturales No Renovables “</u>*** | | | | | | |
| **Actividad o tarea** | **Metodología** | **Fecha** | | **Responsables** | **Presupuesto** | **Resultados esperados** |
| Petición de la Autorización respectiva a Director del Área. | Petición por medio de un Oficio dirigido al Director del Área, y a su vez al Coordinador de Carrera | 25/04/08 | 27/04/08 | Egdo. Darwin Calle.<br><br>Egdo. Daniel Quezada | $ 10 | Información de cómo agregar nuevos componentes al IDE de programación<br><br>Netbeans. |
| Implementación de la Herramienta | Pruebas respectivas a los objetivos planteados. | 28/04/08 | 30/04/08 | Egdo. Darwin Calle.<br><br>Egdo. Daniel | | Lograr el acoplamiento satisfactorio de |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | Quezada | | nuestros componentes al IDE de programación Netbeans. |

# Anexo A2: Cronograma Actividades

## 5.0.- BIBLIOGRAFIA

### Sitios Web.-


www.codigolibre.com

www.comunity.java.net

www.google.com

www.java.com

www.lawebdelprogramador.com

www.netbeans.org

www.opensourcejdk.com

www.programacion-castellano.com

www.sun.com

www.toptutoriales.com

**ANEXO 3: Formato de Encuesta.**

# UNIVERSIDAD NACIONAL DE LOJA

**AREA DE ENERGIA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES.**

Con la presenta encuesta dirigida a los señores profesores y estudiantes de la Carrera de Ingeniería en Sistemas se pretende ver el grado de aceptación del Proyecto de Tesis denominado:

**"Diseño e Implementación de una Herramienta Visual que permita la creación de nuevos Componentes SWING utilizando Java 2D Y 3D".**

Nombre de la Aplicación: **Swing Evolution 1.1.0.**

1. **¿Cuál sería su calificación acerca del Ambiente de Desarrollo Swing Evolution 1.1.0 en lo que corresponde a su Entorno Gráfico?**

Excelente ( )

Muy Buena ( )

Buena ( )

Regular ( )

**2. ¿La aplicación le parece amigable al usuario?**

SI ( )

NO ( )

3. **¿Qué tal le parece la distribución de los diferentes controles visuales dentro de la aplicación Swing Evolution 1.1.0?**

Excelente ( )

Muy Buena ( )

Buena ( )

Regular ( )

**4. ¿En cuánto a rendimiento cómo calificaría a la aplicación?**

Rápida ( )

Lenta ( )

Muy Lenta ( )

5. **¿En lo que concierne al tiempo de diseño para la personalización de los componentes Swing como lo calificaría?**

Excelente ( )

Muy Buena ( )

Buena ( )

Regular ( )

**6. ¿Qué tal le parecen los nuevos componentes Swing generados por la aplicación?**

Excelentes ( )

Muy Buenos ( )

Buenos ( )

Regulares ( )

7. **¿Cree que esta aplicación es una buena herramienta de trabajo para el programador, para la creación de interfaces gráficas con una nueva apariencia visual?**

SI ( )

NO ( )

Porque: ……..

……..

**8. ¿Cómo calificaría a cada uno de los componentes generados por la aplicación Swing Evolution en cuanto a su apariencia visual?.**

Excelentes ( )

Muy Buenos ( )

Buenos ( )

Regulares ( )

**9. ¿En la creación de los componentes visuales como cataloga usted a la aplicación Swing Evolution?.**

Excelentes ( )

Muy Buenos ( )

Buenos ( )

Regulares ( )

**10. ¿Cuál sería su calificación en lo referente a la creación y generación de código del componente personalizado en la aplicación?**

Excelentes ( )

Muy Buenos ( )

Buenos ( )

Regulares ( )

**11. ¿La ejecución de la aplicación Swing Evolution en la plataforma Windows como usted la calificaría?**

Excelente ( )

Muy Bueno ( )

Bueno ( )

Regular ( )

**12. ¿La ejecución de la aplicación Swing Evolution en la plataforma Linux como usted la calificaría?**

Excelente ( )

Muy Bueno ( )

Bueno ( )

Regular ( )

**13. ¿En la ejecución de las tareas de limpieza, compilación y generación de módulos .JAR del proyecto cuál es su calificación?**

Excelente ( )

Muy Bueno ( )

Bueno ( )

Regular ( )

**Gracias por su colaboración**